Software Design Specification

AI Powered Hospital Information Management System

Project Code:

HIMS-AI-2526

Internal Advisor:

Mr. Aamir Zia

Project Manager:

Dr. Muhammad Ilyas

Project Team:

Ali Akbar	BSCS51F22R036	(TL)
Muhammad Najee Ullah Noon	BSCS51F22S043	

Submission Date:

Dec 15,2025

_____

Project Manager's Signature

Document Information

Category	Information
Customer	Computer Science , University Of Sargodha
Project	<Hospital Information Management System >
Document	Requirement Specifications
Document Version	1.0
Identifier	HIMS-AI-2526
Status	Draft
Author(s)	Ali Akbar Muhammad Najee Ullah Noon
Approver(s)	PM
Issue Date	Dec. 15, 2025
Document Location	
Distribution	1. Advisor 2. Project Manager 3. Project Office

Definition of Terms, Acronyms and Abbreviations

This section should provide the definitions of all terms, acronyms, and abbreviations required to interpret the terms used in the document properly.

Term	Description
HIMS	Hospital Information Management System
RS	Requirements Specifications
HIPPA	Health Insurance Portability and Accountability Act
GDPR	General Data Protection Regulation
LAN	Local Area Network
CNIC	Computerized National Identity Card

Table of Contents

1. Introduction

1.1 Purpose of Document

The document presents the design requirements of the project "AI Powered Hospital Information Management System". It gives a detailed explanation of system's design, functionality and how the AI-based solution will automate the patient data management and streamline hospital workflows. Audience covers the project stakeholders, developers, healthcare IT managers and hospital administrators. It serves as reference to ensure that all parties have a unified understanding of the system's objectives, architecture, constraints and operational behaviors. The document also ensures that the design aligns with HIPPA and GDPR. It will act as a guide during development, testing and deployment phases ensuring that the final product fulfills its intended purpose of reducing administrative burden and improving hospital efficiency.

1.2 Project Overview

Modern hospitals often struggle with inefficiencies caused by manual data entry, fragmented record-keeping, and time-consuming administrative processes. These challenges lead to delays in patient care, errors in documentation, and an increased workload on medical staff.

To address these issues, the AI-Powered Hospital Information Management System (HIMS) is proposed as an intelligent, automated platform that integrates speech recognition and natural language processing to streamline hospital workflows.

The system functions as follows:

1. **AI-Based Conversation Recording and Analysis:**
   The system listens to doctor–patient interactions in real time, identifies the speaker, and extracts key medical information such as symptoms, diagnosis, prescribed treatments, and follow-up details.

2. **Doctor Confirmation and Data Storage:**
   Before saving, the extracted data is displayed to the doctor for verification. Once confirmed, it is automatically stored in a centralized hospital database.

3. **Dedicated Dashboards:**

- **Doctor Dashboard:** Enables doctors to view patient histories, review notes, and manage consultations.

- **Admin Dashboard:** Allows hospital administrators to manage records, monitor data flow, and ensure system compliance.

- **Patient Portal:** Provides patients with access to their medical history, prescriptions, and reports.

The core objective of this project is to reduce administrative workload, improve record accuracy, and enable real-time access to patient information. By leveraging advanced technologies such as Whisper for speech-to-text processing, GPT-based models for data extraction, and PostgreSQL

for secure data management, the proposed system enhances hospital efficiency and promotes the adoption of AI-driven healthcare solutions.

1.3 Scope

The scope of the AI-powered Hospital Information Management System (HIMS) is defined by its focus on improving patient record management, reducing administrative workload, and supporting accurate medical documentation through AI assistance, while keeping doctors in control of validation. The project is specifically designed for healthcare institutions and has clear boundaries on what it will and will not cover.

Inclusions:

- **AI-Assisted Medical Note Creation:** The system captures doctor-patient conversations, highlights relevant medical details (symptoms, diagnoses, tests), and presents them for doctor confirmation before storing in the database.
- **Centralized Patient Record Management:** Hospitals can maintain structured medical histories, accessible to authorized staff across visits.
- **Integrated Staff Dashboard:** Enables staff to verify or edit patient records, and oversee hospital operations such as scheduling and reporting.
- **Secure Patient Access:** Patients can receive their reports digitally through their profile or via staff-provided printed documents.
- **Data Analytics for Hospitals:** Aggregated data helps identify trends in patient issues, treatment effectiveness, and administrative workload.

Exclusions:

- **Full Clinical Decision-Making:** The system does not recommend treatments or prescribe medicines; it only records and structures doctor-validated information.
- **Cross-Hospital Data Sharing:** Patient data will not automatically transfer between hospitals unless formally integrated with external systems.
- **Self-Registration by Patients:** Initial patient accounts will be created by hospital staff; self-service registration will not be included in the current scope.
- **Non-Medical Hospital Operations:** The system will not handle unrelated tasks such as payroll, pharmacy stock management, or facility maintenance.

2. Design Considerations

2.1 Assumptions and Dependencies

Assumptions

- The hospital's medical staff and administrative personnel will cooperate fully in adopting the new HIMS platform and provide accurate input data during initial setup and daily usage.
- The hospital's existing hardware infrastructure (computers, microphones, and secure servers) will meet the minimum system requirements for smooth operation.
- Internet connectivity within the hospital premises will remain stable to enable real-time data synchronization and AI model processing.
- Doctor-patient conversation recordings will be conducted in environments with minimal background noise to ensure the accuracy of the speech-to-text and entity extraction modules.
- All users—including doctors, nurses, and administrative staff—will receive basic training on how to use the system effectively.
- The hospital's policies and workflows will remain consistent during the system's development and deployment phases, ensuring alignment between technical and operational procedures.

Dependencies

- The AI modules depend on external speech-to-text and natural language processing APIs for transcribing and extracting key medical data from doctor-patient conversations.
- The system relies on a stable connection to the hospital's central database for data storage and retrieval of patient records, diagnoses, and treatment histories.
- Regular data updates and maintenance must be performed by authorized IT personnel to ensure accuracy and prevent inconsistencies.
- Integration with existing hospital modules (e.g., billing, pharmacy, and lab systems) depends on the availability and compatibility of their APIs or export functionalities.
- The system's deployment and maintenance depend on compliance with healthcare regulations such as HIPAA or local data protection laws to ensure security and confidentiality.
- Timely software updates, API key renewals, and support from external AI service providers are necessary to maintain full functionality and performance.

2.2 Risks and Volatile Areas

1. Integration with External Systems

Risk: Need to integrate with Insurance APIs, ASR engines, NLP engines, etc. Protocols may change.
Design Response:

- Use API gateway + adapter layer.
- Defined integration contracts with versioning.
- Decouple external systems with message queues.

2. Technology and Infrastructure Changes

Risk: Upgrading databases, switching cloud services, or adopting new frameworks can disrupt the system.
Design Response:

- Use containerized deployments (Docker/K8s).
- Abstract data access using repository pattern.
- Stateless backend services where possible.

3. Performance and Scalability Demands

Risk: Increased patient volume, higher concurrency, real-time reports, heavy NLP/ASR workloads.
Design Response:

- Auto-scaling for API services.
- Caching layers for frequent reads.
- Queue-based processing for heavy modules (ASR, NLP).

4. Security Threats

Risk: Cyberattacks, unauthorized access, ransomware, misuse of patient data.
Design Response:

- Role-based + attribute-based access control.
- Encryption of data in transit + at rest.
- Continuous monitoring and vulnerability scans.

5. Third-Party Service Dependency Risks

Risk: ASR/NLP services, SMS gateways, cloud storage may change pricing, APIs, or availability.
Design Response:

- Use abstraction/interface layer.

- Fallback providers (backup ASR, backup SMS).
- Circuit breakers and retries.

6. Operational Risks

Risk: Server downtime, network issues, database failure.
Design Response:

- Redundancy: master-slave DB, load balancer.
- Backup + disaster recovery plan.
- Health monitoring + alerting.

3. System Architecture

3.0 Use Case Diagram

Figure 3.0 Use Case Diagram

3.1 System Level Architecture

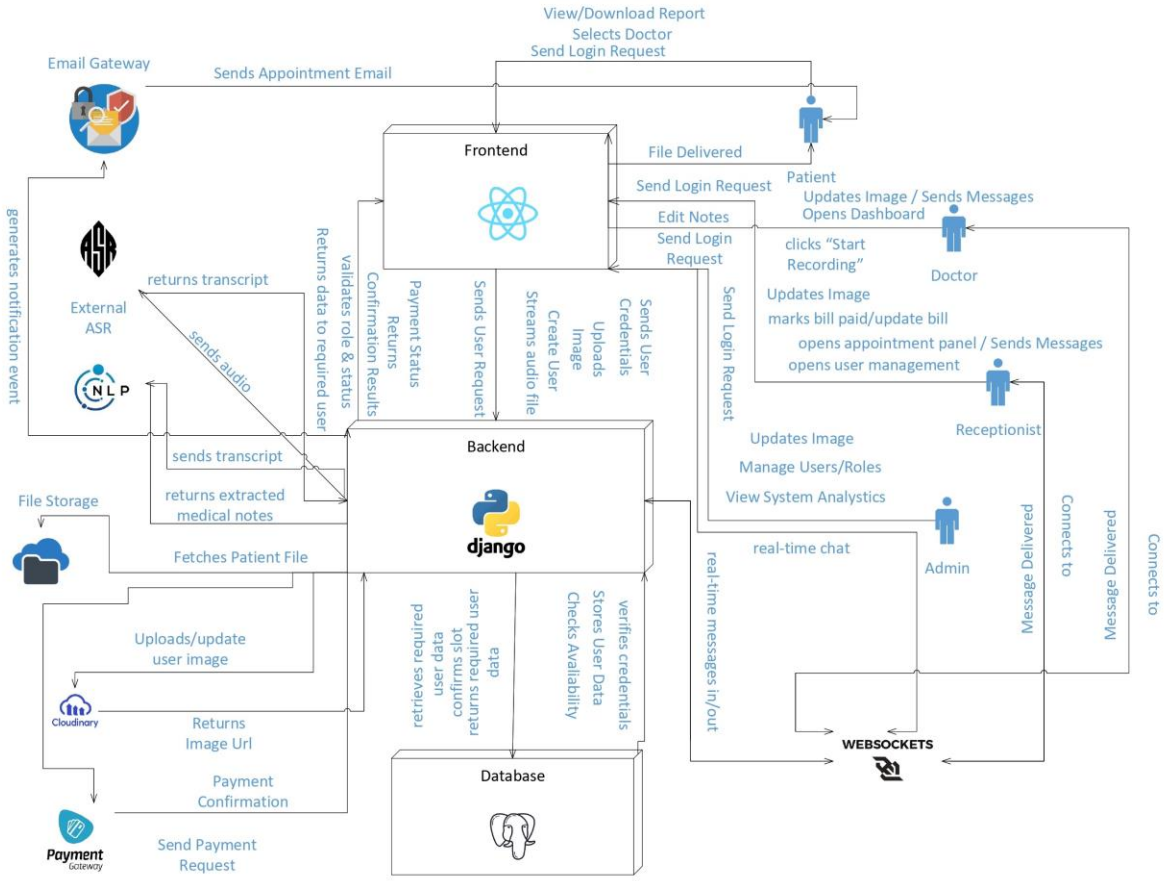


Figure 3.1 System Architecture Diagram

3.2 Sub-System / Component / Module Level Architecture

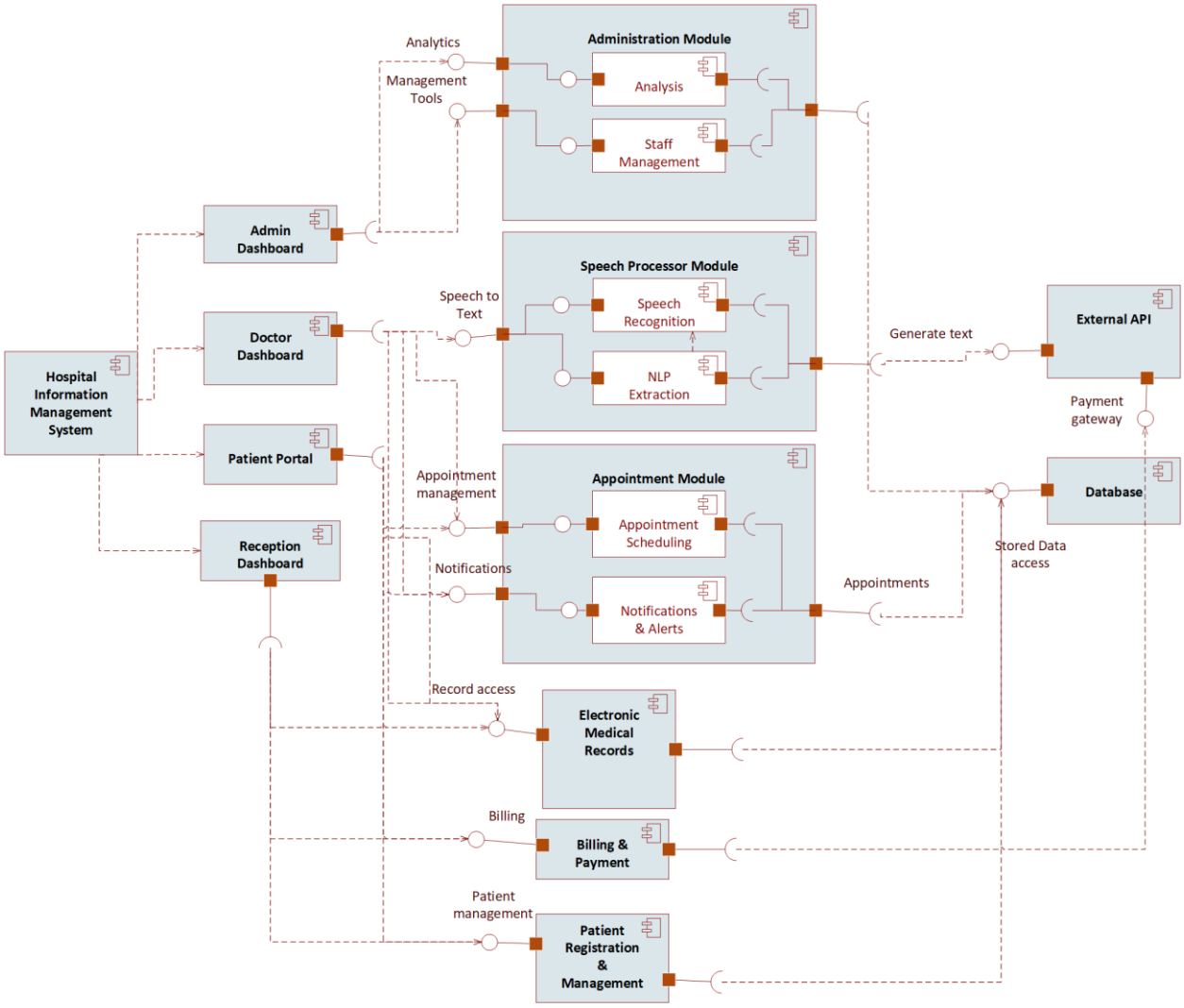


Figure 3.2 Component Diagram

3.3 Sub-Component / Sub-Module Level Architecture (1…n)

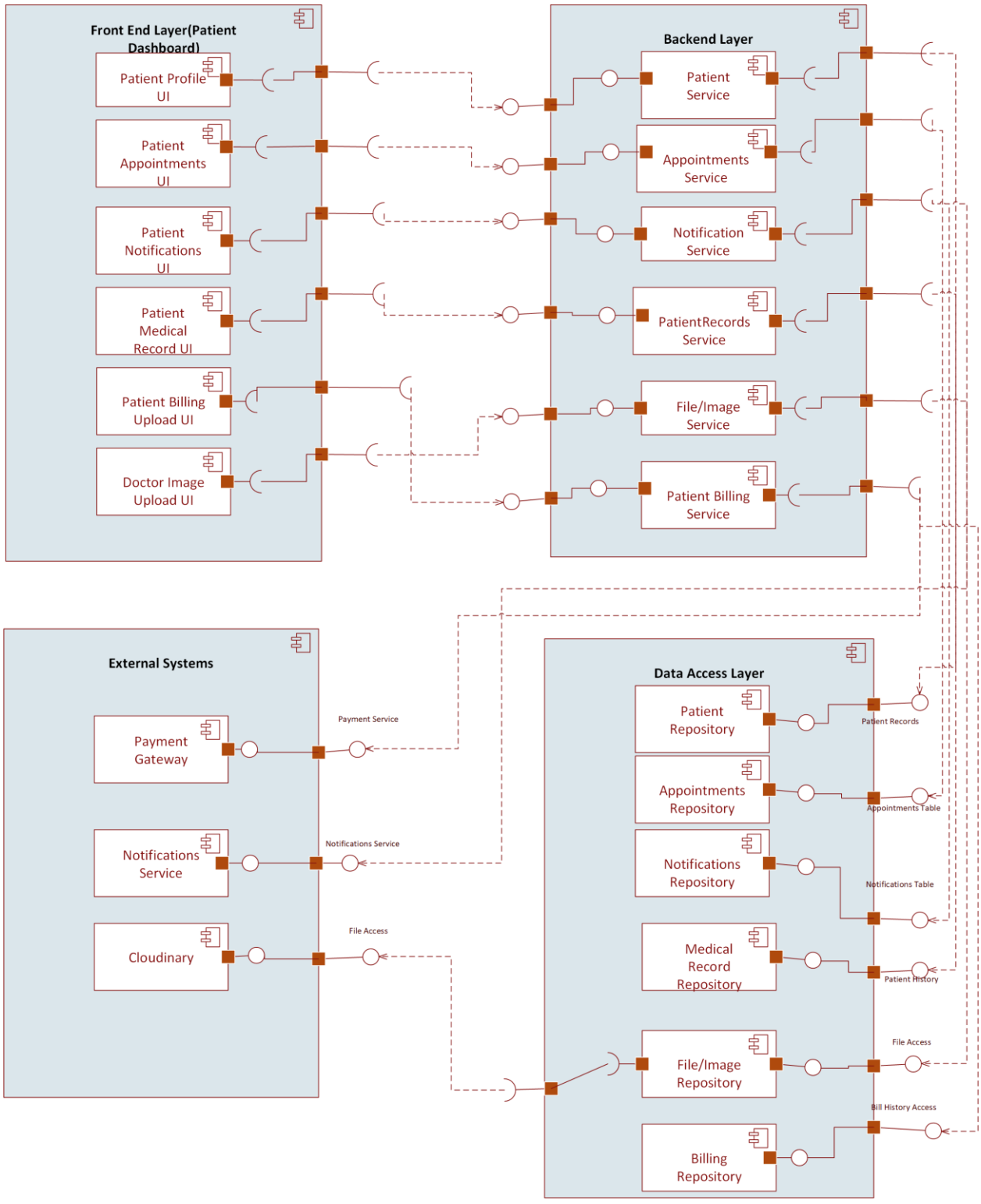


Figure 3.3.1 Patient Component Diagram

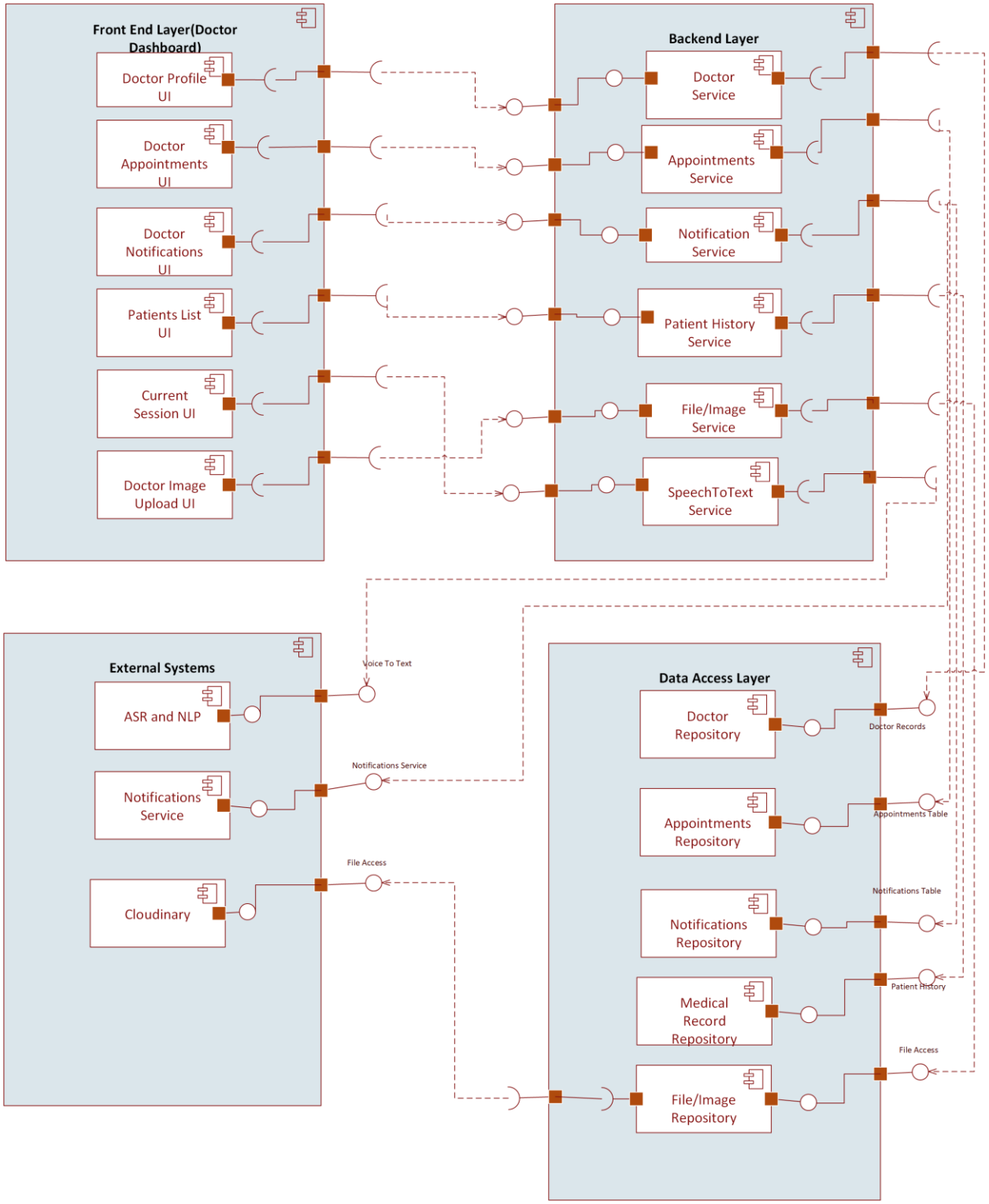


Figure 3.3.2 Doctor Component Diagram

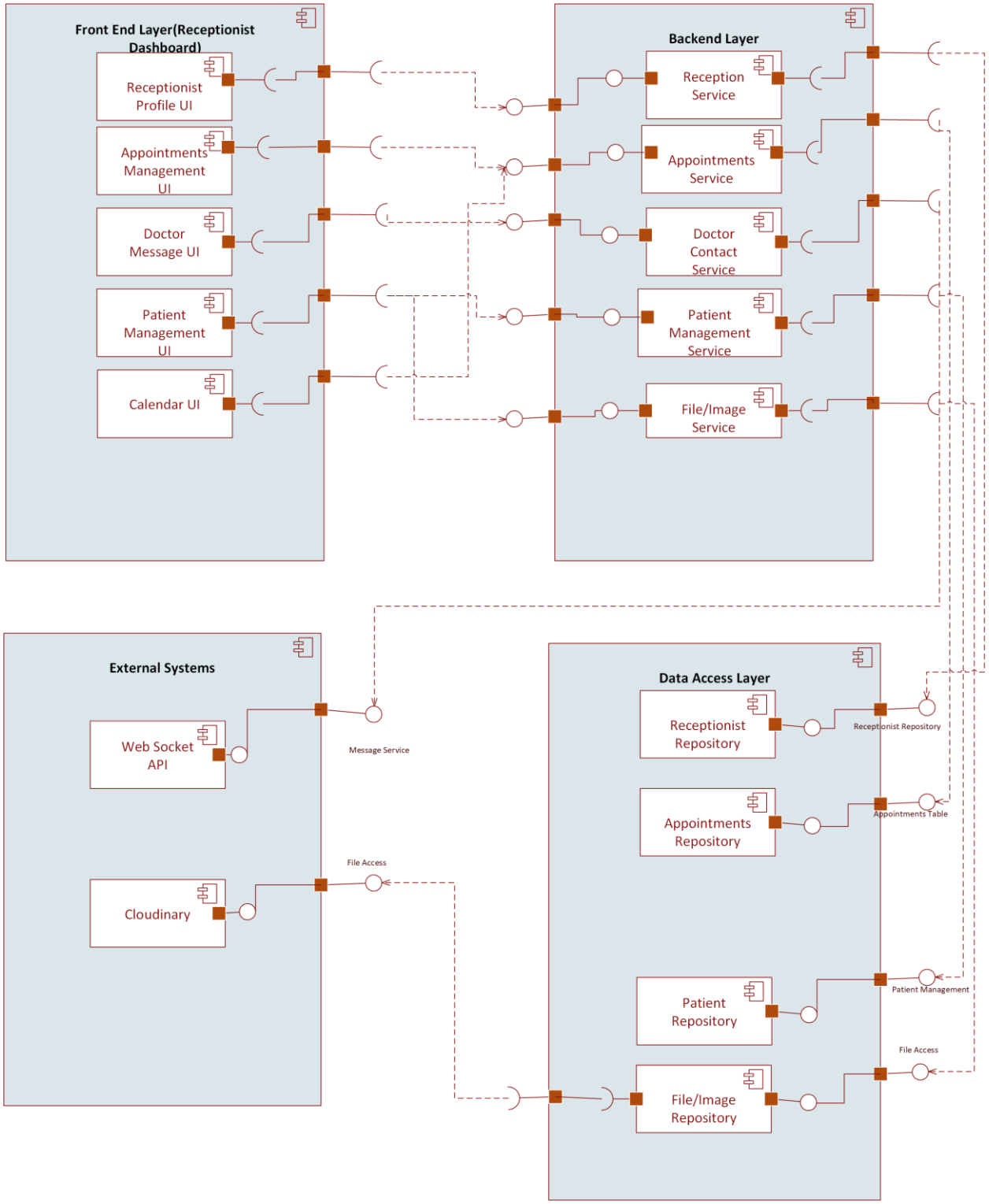


Figure 3.3.3 Receptionist Component Diagram

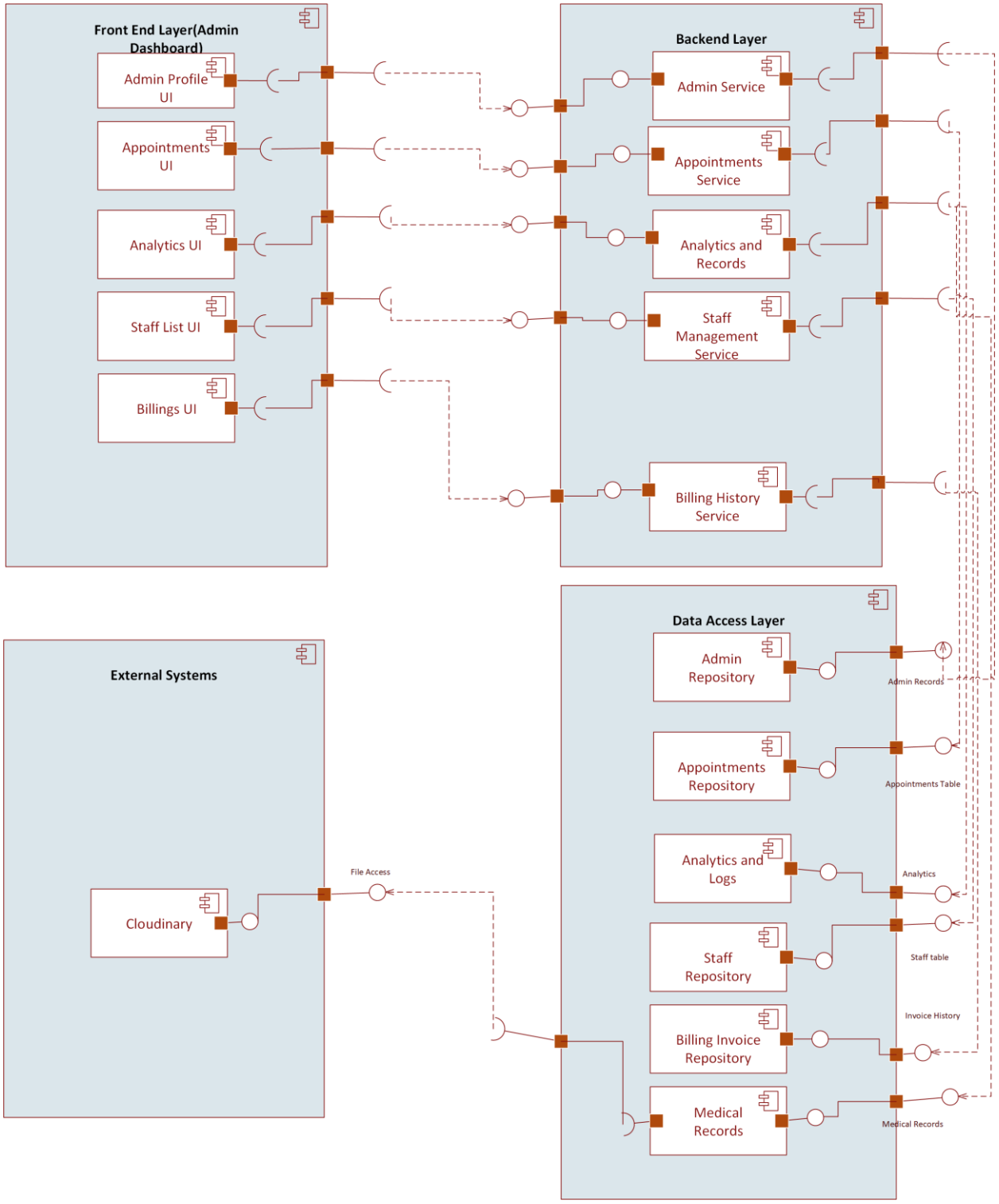


Figure 3.3.4 Admin Component Diagram

4. Design Strategies

4.1 Strategy 1: Modular Layered Architecture

The system is structured into logical layers:

1. **Presentation Layer (Frontend UI)**
   Patient portal, doctor portal, receptionist portal and admin dashboards.
2. **Application Layer (Backend Logic)**
   Appointment handling, patient record management, report processing, ASR pipelines.
3. **Data Access Layer**
   Repository classes, ORM queries, database communication.
4. **Database Layer**
   PostgreSQL for structured hospital data.

Reasoning

- Separates UI, business logic, and database interaction.
- Makes each module easy to modify or extend.
- Reduces complexity and avoids tangled dependencies.

Trade-offs

- Requires discipline to maintain layer boundaries.
- Slight overhead in communication between layers.

If the hospital later introduces **pharmacy**, **lab integration**, or **insurance**, new modules plug into the architecture without affecting existing ones.

4.2 Strategy 2: API-Driven / Service-Oriented Design

All modules communicate using clearly defined **REST APIs**.

Examples:

- `/appointments/book`
- `/patients/update-history`

Reasoning

- Clean separation between frontend and backend.
- Easy to integrate with mobile apps, external labs, or third-party hospital devices.

- Allows independent scaling of services later.

Trade-offs

- More API design work initially.
- Network latency between services.

The **voice-to-text ASR service** can run separately without modifying the main backend. This makes the system future-proof for new AI modules.

4.3 Strategy 3: Future System Extension & Plug-in Architecture

The architecture is designed to allow **new workflows and modules** without rewriting existing ones. Strategies used:

- Interface-based design
- Configurable modules
- Separate routing for each domain (patients, appointments, diagnosis, reports)

Reasoning

Hospitals frequently add:

- New lab machines
- New departments
- New data flows (insurance, pharmacy automation)

Trade-offs

- Requires abstraction and planning during early design.
- Some modules might have more configuration overhead.

If we add a **Radiology module**, it simply becomes:

- New routes: `/radiology/upload`, `/radiology/report`
- New table: `radiology_reports`
- No impact on appointments, patients, or ASR pipeline.

4.4 Strategy 4: Unified Data Management Strategy

The system uses a consistent approach to data storage:

- **PostgreSQL** for structured hospital records
- **Cloud storage** for files (images, audio)

- **ACID transactions** for sensitive updates
- **Foreign keys** to prevent inconsistent data
- **Indexed queries** for real-time performance (appointments, patient history)

Reasoning

- Ensures correctness, especially for medical data.
- Enables fast retrieval of patient histories.
- Supports large-scale hospital operations.

Trade-offs

- Requires schema planning.
- Some features (e.g., full-text search) need extra tools.

Appointment booking uses transactions to prevent **double-booking** when multiple receptionists try to schedule the same slot at the same time.

4.5 Strategy 5: Concurrency & Synchronization Controls

Some hospital actions need safe handling of parallel requests:

- Appointment booking
- Prescription updates
- Audio file uploads → ASR processing
- Simultaneous doctor and admin updates on patient record

Concurrency mechanisms used:

- Database locks
- Transaction isolation
- Atomic operations
- Queues for long tasks (ASR, NLP)

Reasoning

- Guarantees consistent data.
- Prevents race conditions.
- Critical for time-sensitive medical operations.

Trade-offs

- High concurrency may reduce performance due to locking.
- Requires careful backend design.

When two receptionists book the same doctor slot:

1. Backend locks the slot record
2. Checks availability
3. Inserts appointment only if slot is free
4. Releases the lock

This ensures **only one** booking is stored.

4.6 Strategy 6: Reusability of Components

Common utilities are designed for reuse across modules:

- Logging service
- Authentication middleware
- File upload utility
- Notification/email service
- Validation helpers

Reasoning

- Reduces repeated code.
- Makes maintenance easier.
- Ensures consistent behavior across modules.

Trade-offs

- Shared services must be carefully versioned to avoid breaking modules.

The **file upload service** (used by voice recordings and lab reports) is a single shared component used by multiple modules.

4.7 Strategy 7: Secure-by-Design Architecture

Security is built into the architecture instead of added later:

- Role-based access control (Admin, Doctor, Patient)

- Encrypted communication (HTTPS)
- Input validation
- Audit logging
- Secure authentication tokens

Reasoning

Hospitals handle extremely sensitive patient data.

Trade-offs

- Adds development overhead
- Requires periodic updates and policy checks

Doctors can only access their assigned patient records. Patients cannot view internal notes or doctor comments.

5. Detailed System Design

5.1 Class Diagram

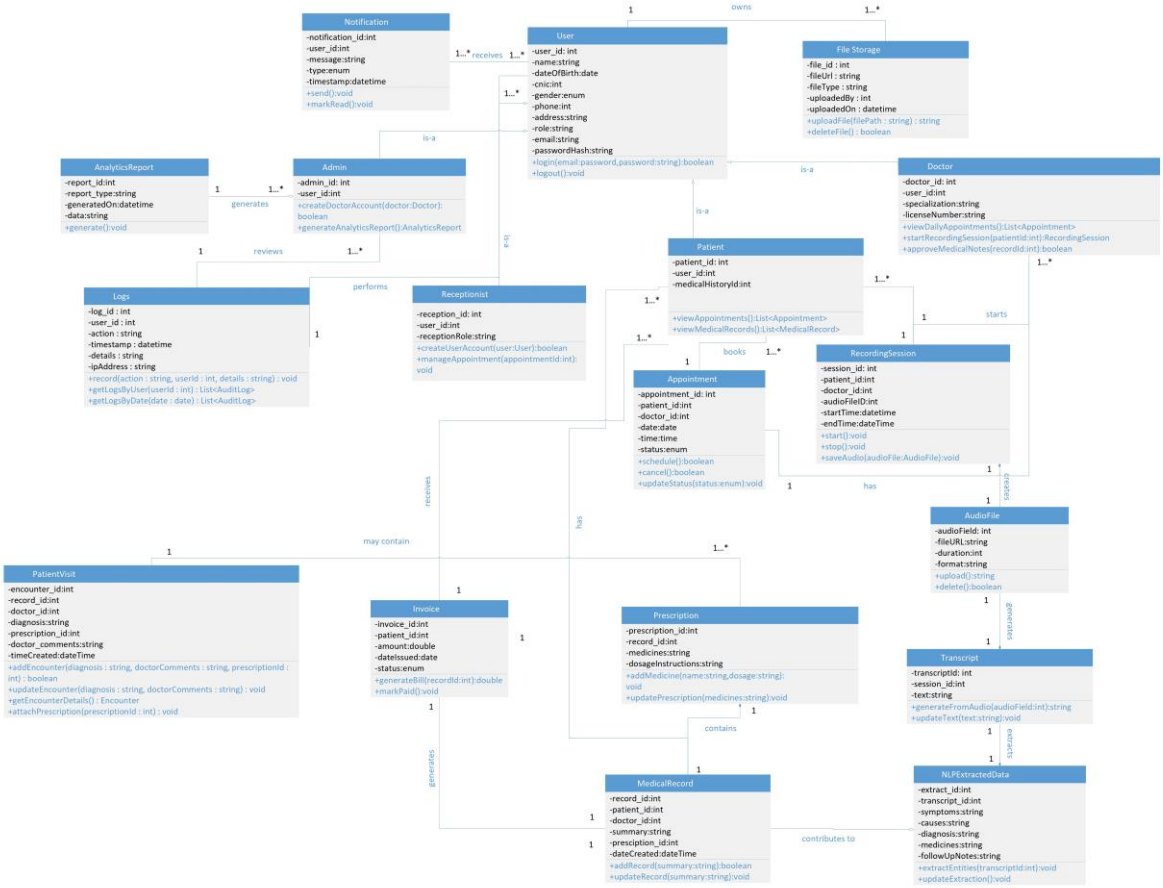


Figure 5.1 Class Diagram

5.2 ER Diagram

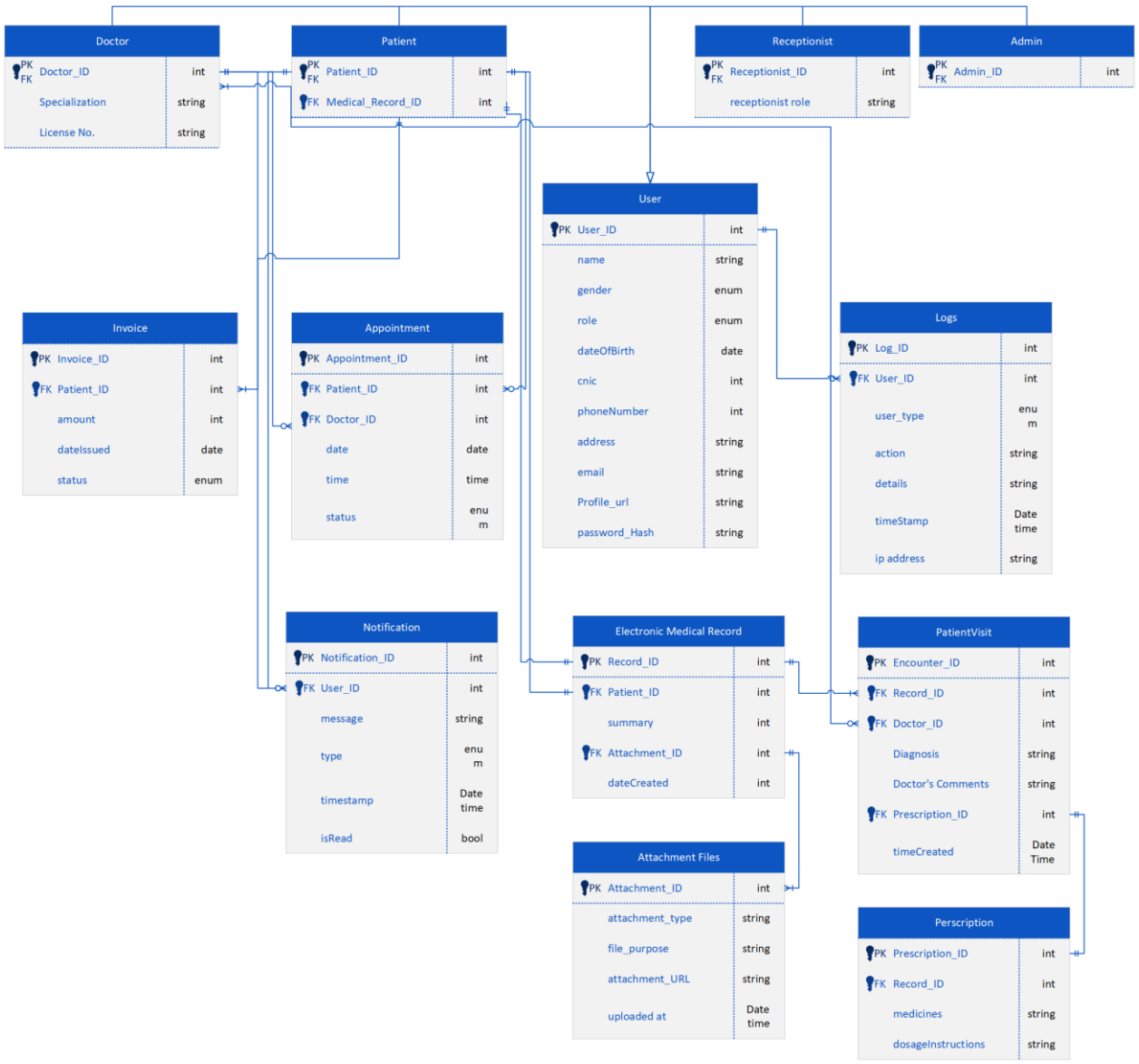


Figure 5.2 ER Diagram

5.3 State Transition Diagram

Figure 5.3 State Transition Diagram

5.4 Sequence Diagram

5.4.1 Sequence Diagram (Login)

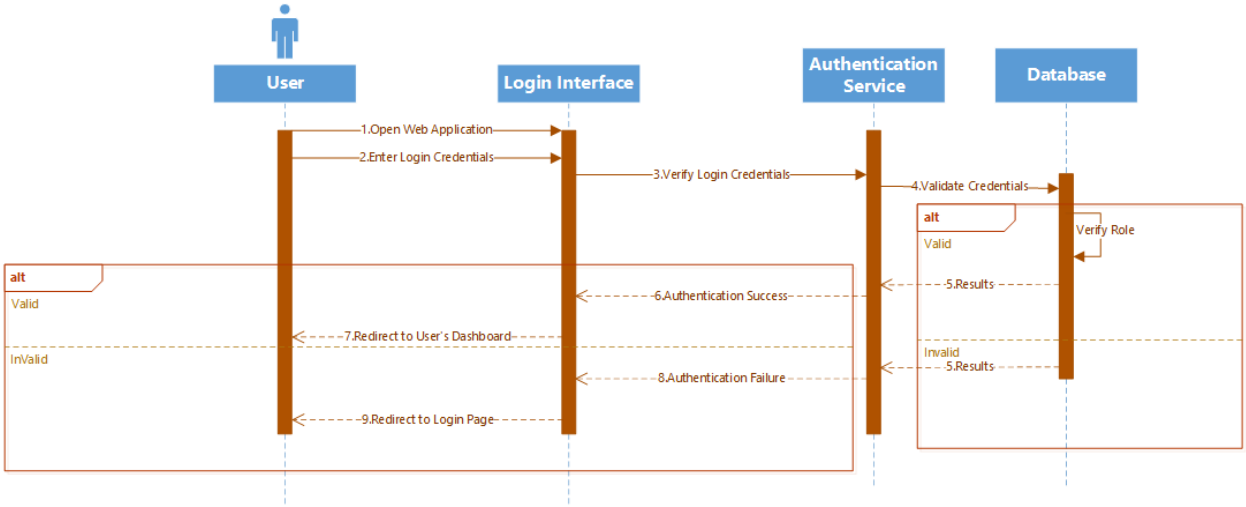


Figure 5.4.1 Sequence Diagram (Login)

5.4.2 Sequence Diagram (Patient)

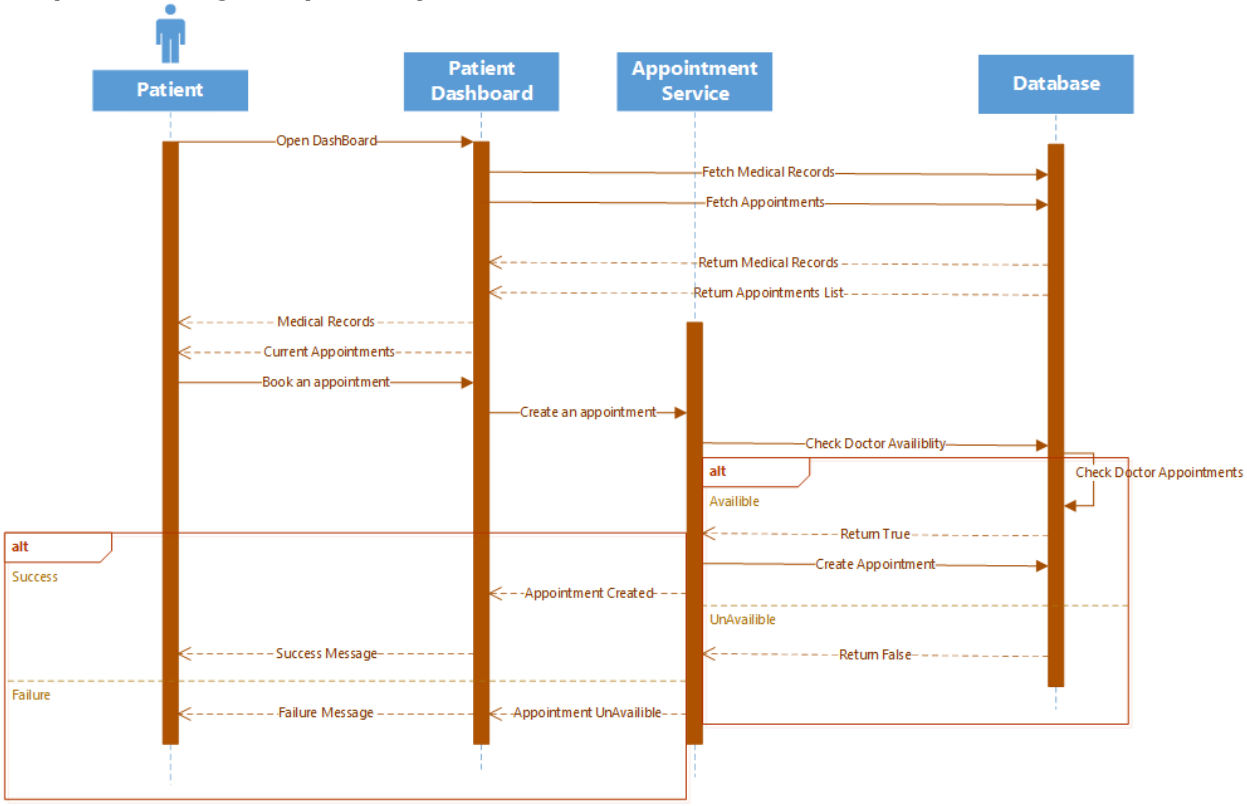


Figure 5.4.2 Sequence Diagram (Patient)

5.4.3 Sequence Diagram (Doctor)

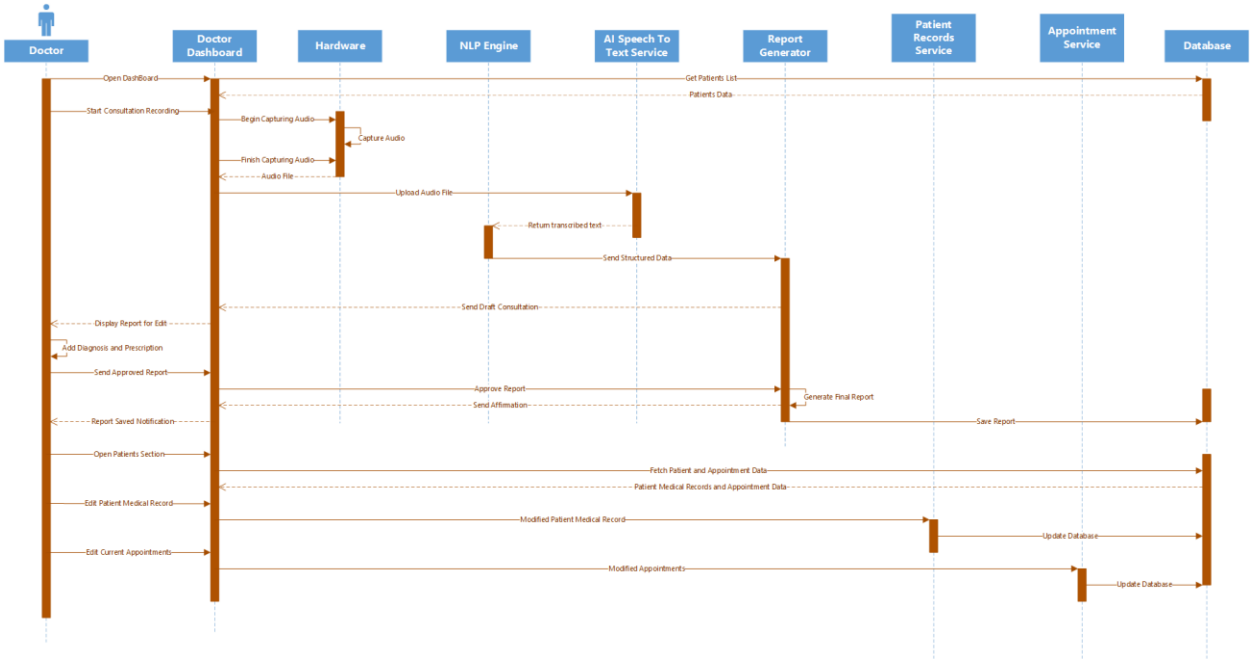


Figure 5.4.3 Sequence Diagram (Doctor)

5.4.4 Sequence Diagram (Receptionist)

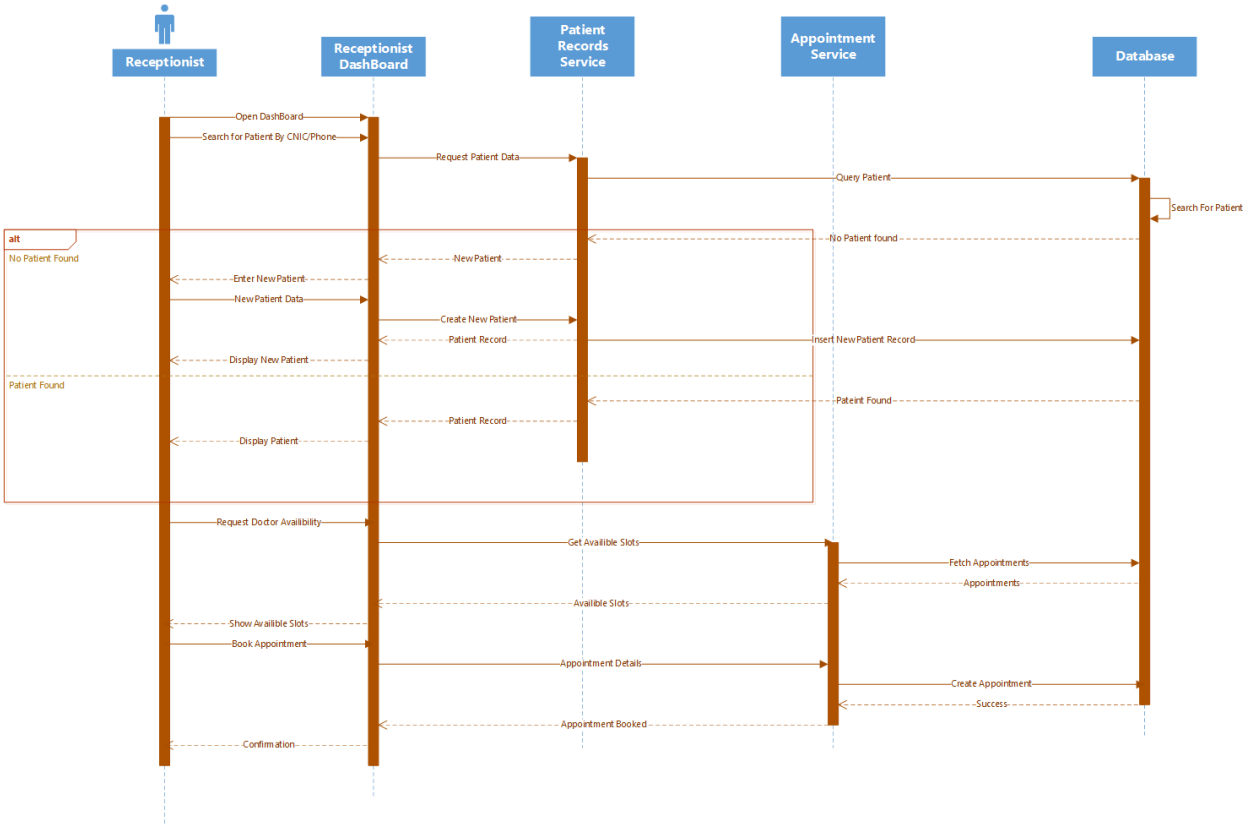


Figure 5.4.3 Sequence Diagram (Receptionist)

5.4.5 Sequence Diagram (Admin)

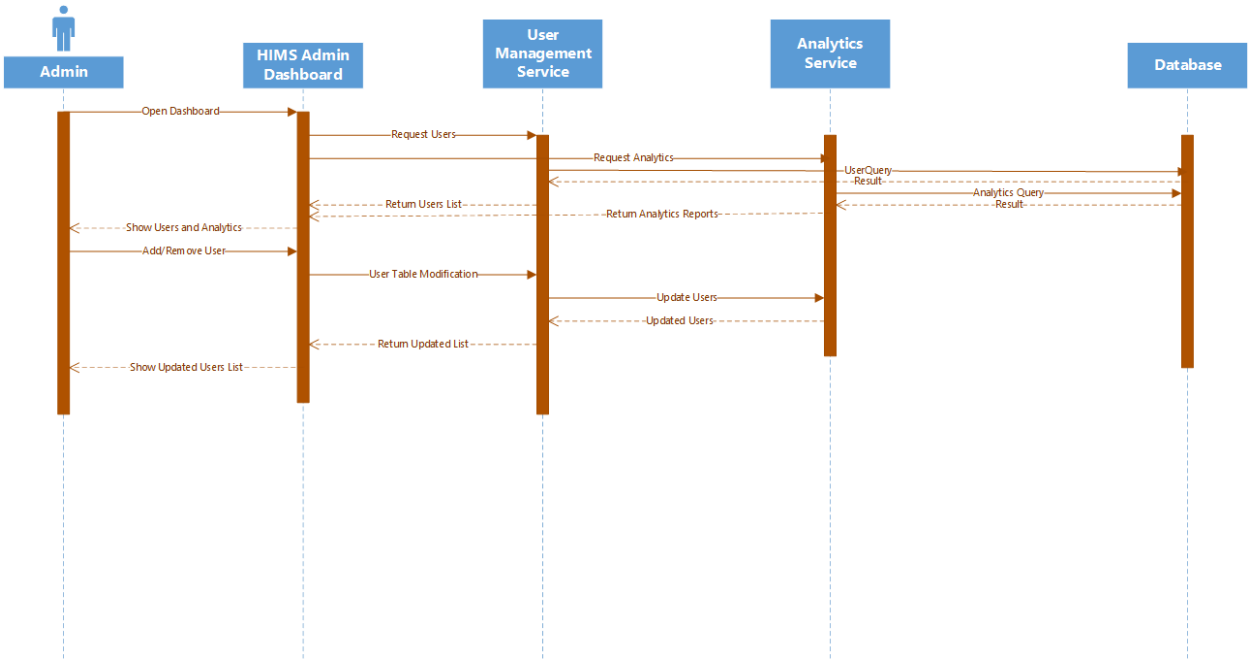


Figure 5.4.5 Sequence Diagram (Admin)

5.5 Data Flow Diagram

5.5.1 DFD Level 0

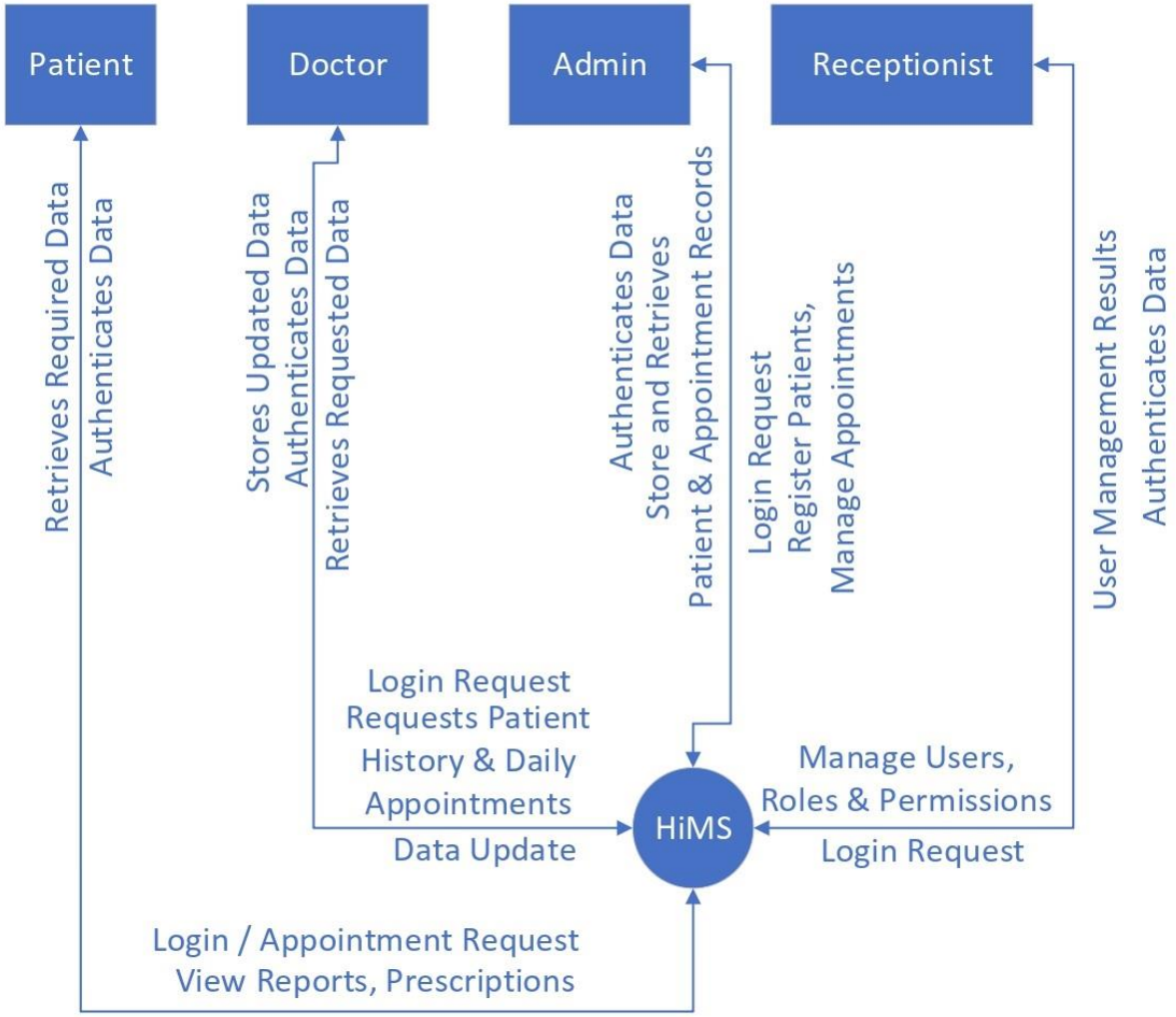


Figure 5.5.1 DFD Diagram (Level 0)

5.5.2 DFD Level 1

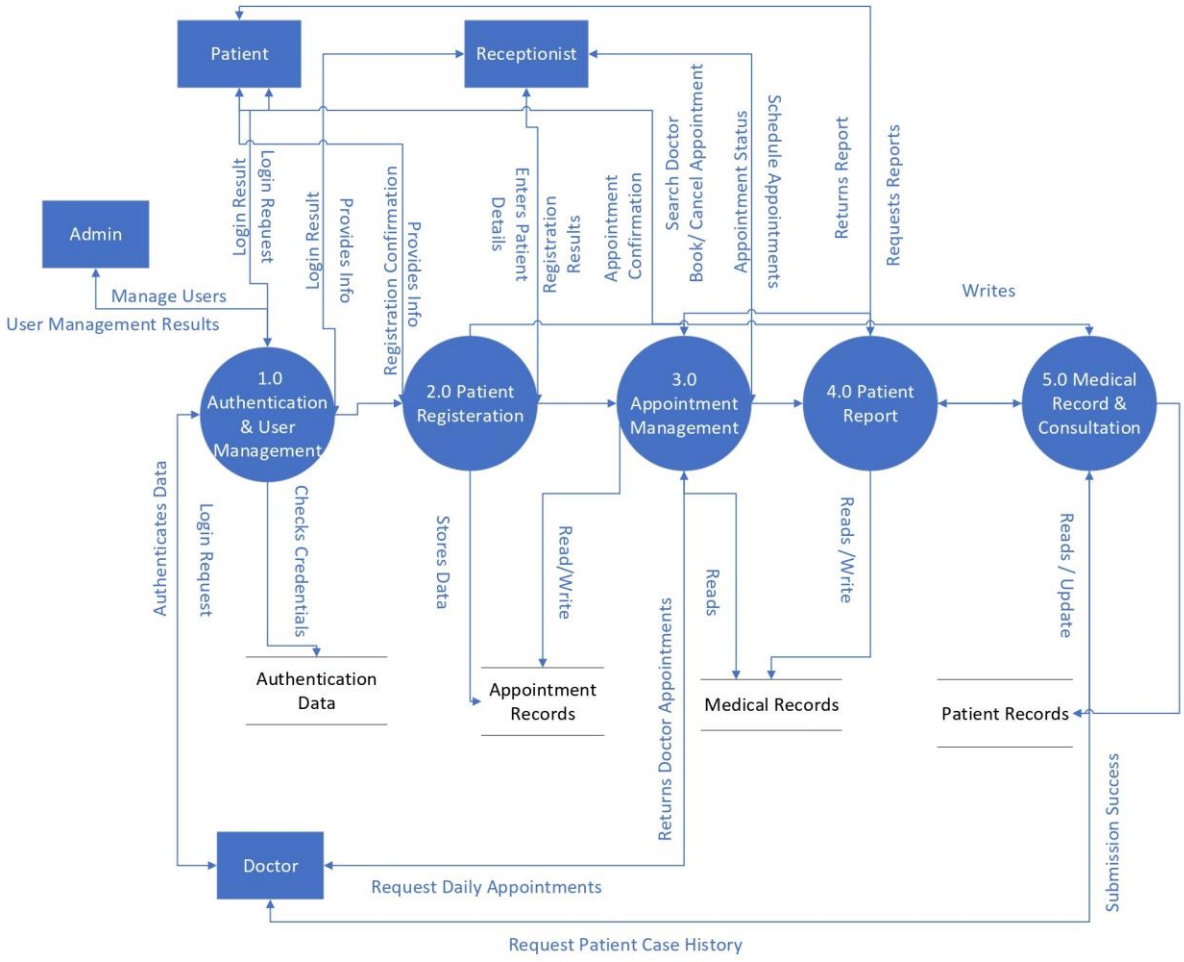


Figure 5.5.2 DFD Diagram (Level 1)

5.5.3 DFD Level 2

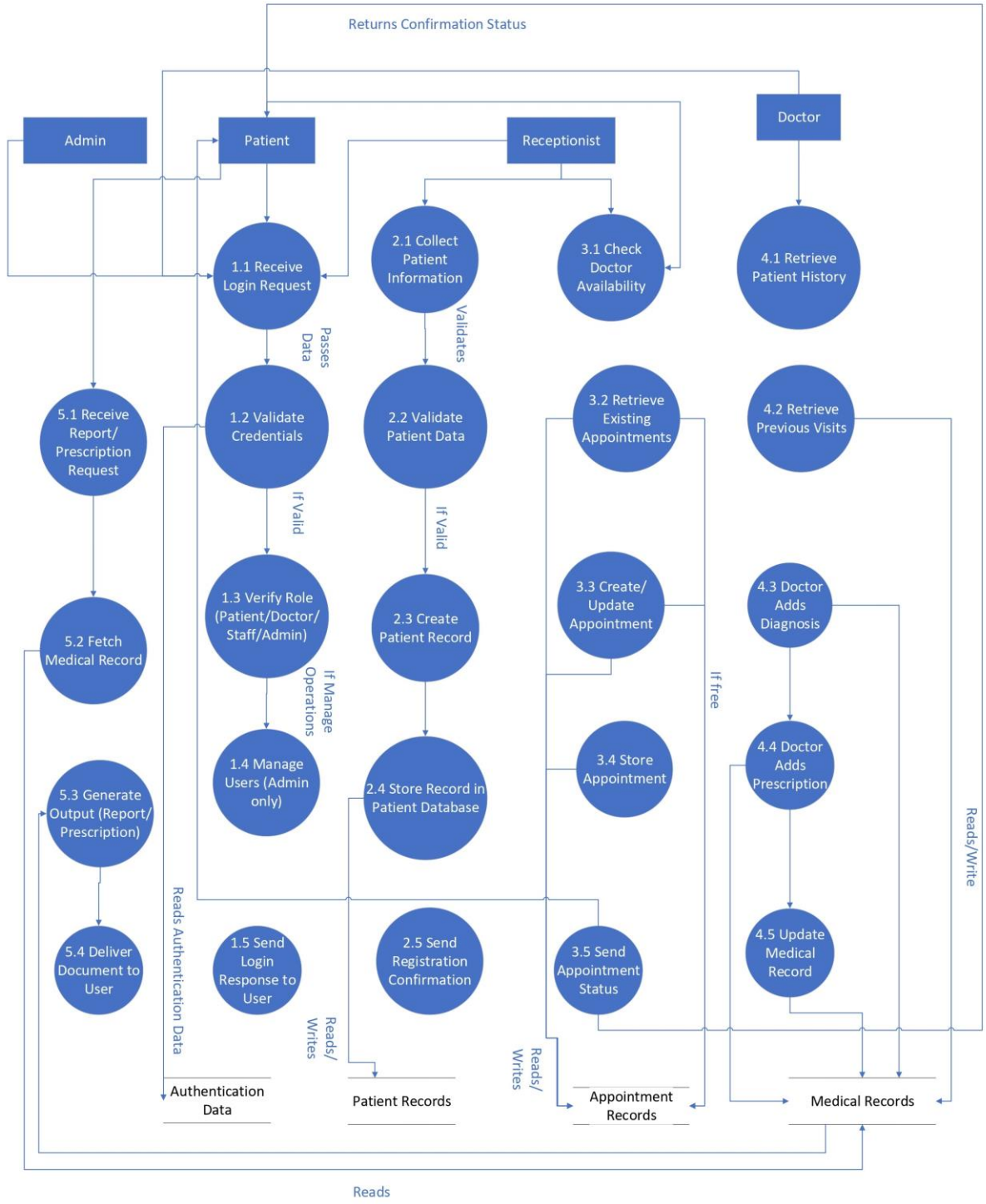
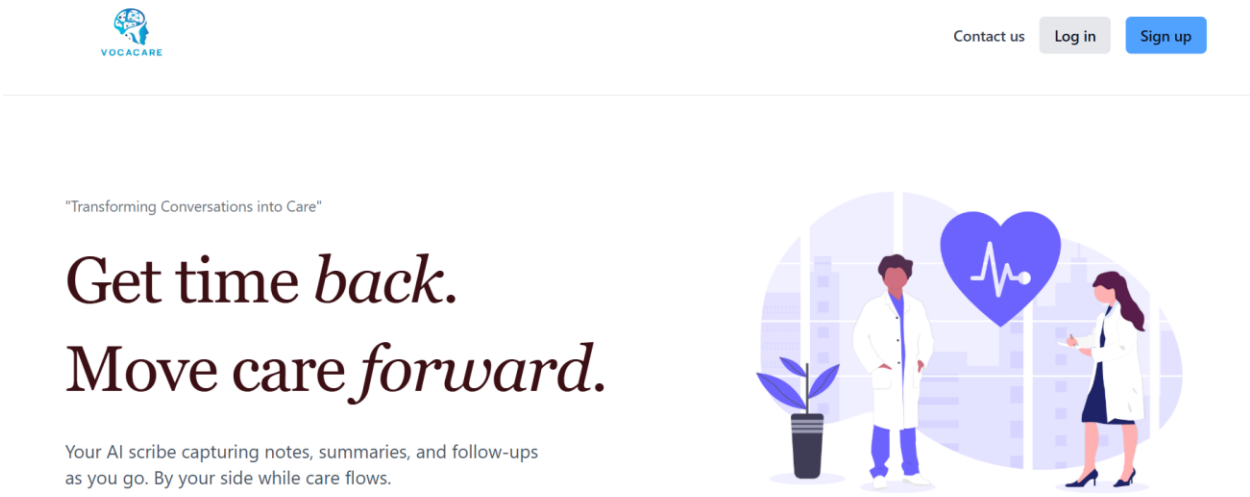


Figure 5.5.3 DFD Diagram (Level 2)

5.6 GUI

5.6.1 Landing Page



Purpose & Context:

- Entry point to the system for all user roles.
- Introduces the platform and leads users to login or signup.

Element Details:

1.Branding Area

- Vocacare logo
- Tagline: “Transforming conversations into care.”

2.Primary Actions

- **Sign In Button** → navigates to Sign in page.
- **Log In Button** → navigates to Login page.

User Flow (Landing Page)

1. User clicks “Sign In”

System Action:

- Redirects to Register Page.

2. User clicks “Log In”

System Action:

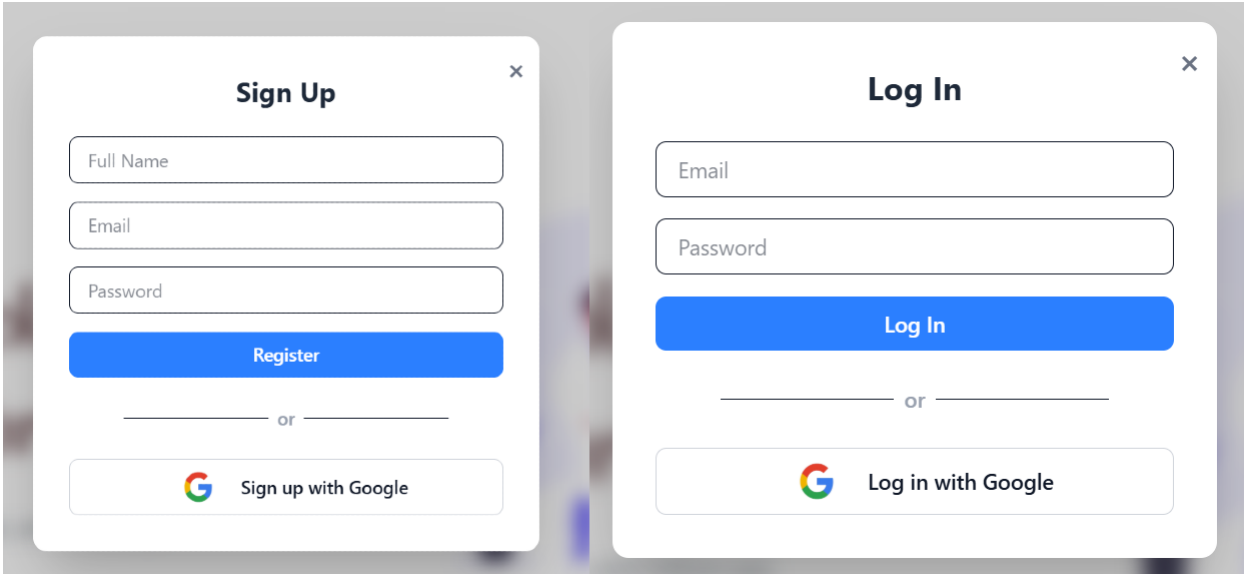
- Redirects to Login Page.

3. User returns to landing

System Action:

- No state stored; fresh page reload.

5.6.2 Register and Login Page



Purpose & Context

Secure access to the system for all roles.

Element Details

Login Form

- Email input
- Sign In button
- Error messages
- Log in with Google

Register Form

- Email input
- Name input
- Password input
    - Register Button

User Flow

1. Enter Email + Password → Click Sign In

System Actions:

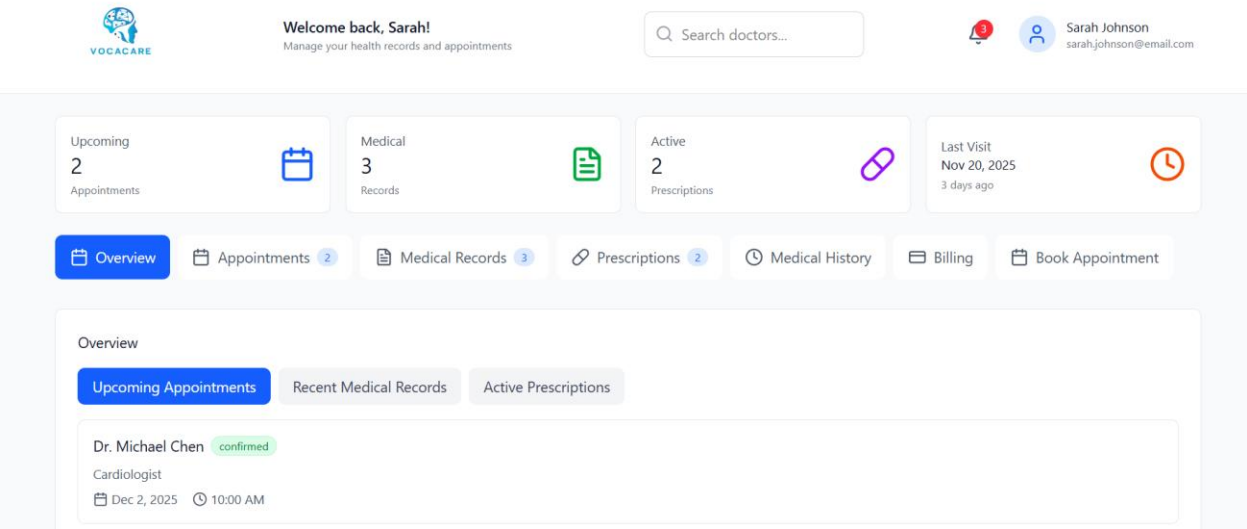
- Validate email format
- Validate password length
- Send credentials to API
- If valid → redirect to relevant dashboard
- If invalid → show inline error: “Incorrect email or password.”

2. Enter Name+ Email + Password → Click Register

System Actions:

- Validate email format
- Validate password length
- Send credentials to API
- If valid → redirect to relevant dashboard
- If invalid → show inline error: “Invalid email or password.”

5.6.3 Patient Dashboard



Purpose & Context

Allow patients to manage bookings, medical records, prescriptions, invoices.

Element Details

Header

- Logo
- Welcome
- Search
- Notifications
- User profile

Modules

- Upcoming Appointments
- Medical History
- Prescriptions
- Billing Summary
- Quick Actions

User Flow

1. Patient books an appointment

System Action:

- Opens calendar
- Select doctor, date, time
- Confirm
- Appointment appears in dashboard

2. Patient views medical history

System Action:

- Load personal medical timeline
- Click entry → opens details

3. Patient pays invoice

System Action:

- Redirect to billing
- Choose payment method
- Confirm payments

4. Patient clicks notification bell

System Action:

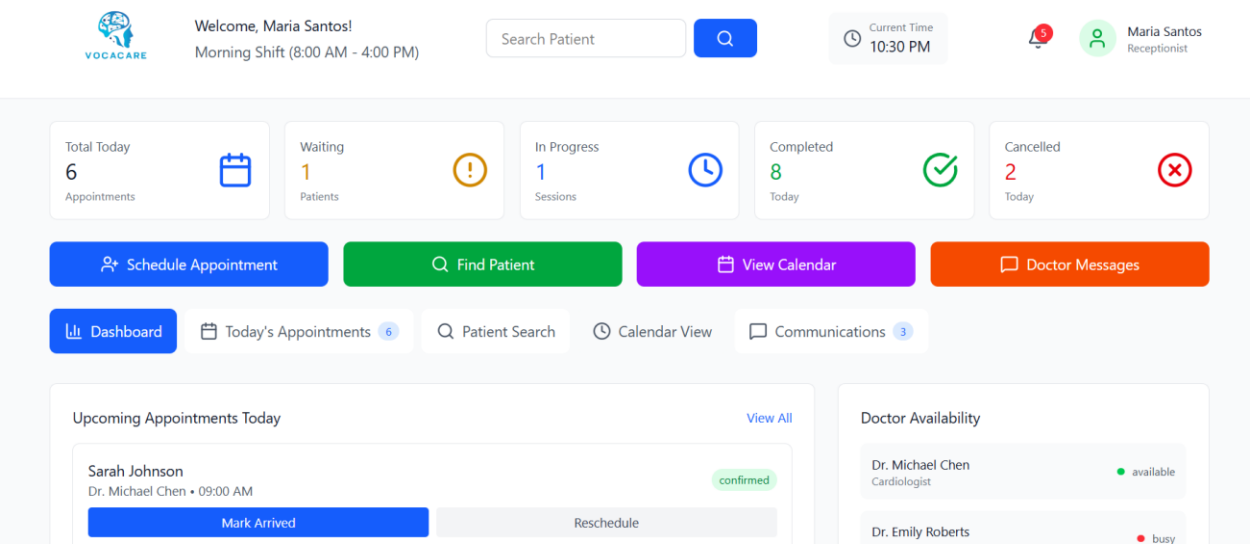
- Dropdown opens
- Notifications marked as read when viewed

5. Patient edits profile

System Action:

- Update profile
- Save and refresh

5.6.4 Receptionist Dashboard



Purpose & Context

Manage patients, check-ins, appointments, scheduling, reception workflow.

Element Details

Header

- Logo
- Welcome section

- Search patients
- Current time display
- Notifications dropdown
- User menu (profile, settings, logout)

Main Modules

- Today's appointments
- Patient search
- Quick actions (Register patient, Book appointment)
- Appointment overview tables

User Flow

1. Receptionist searches for a patient

System Action:

- Query patients
- Show list of matches
- Click result → open patient profile

2. Receptionist clicks "Check-In"

System Action:

- Mark appointment as "Arrived"
- Update doctor's appointment list
- Toast: "Patient checked in."

3. Click "Book Appointment"

System Action:

- Redirect to appointment form
- Pre-fill patient info if patient was selected

4. Clicking Notifications Icon

System Action:

- Dropdown opens

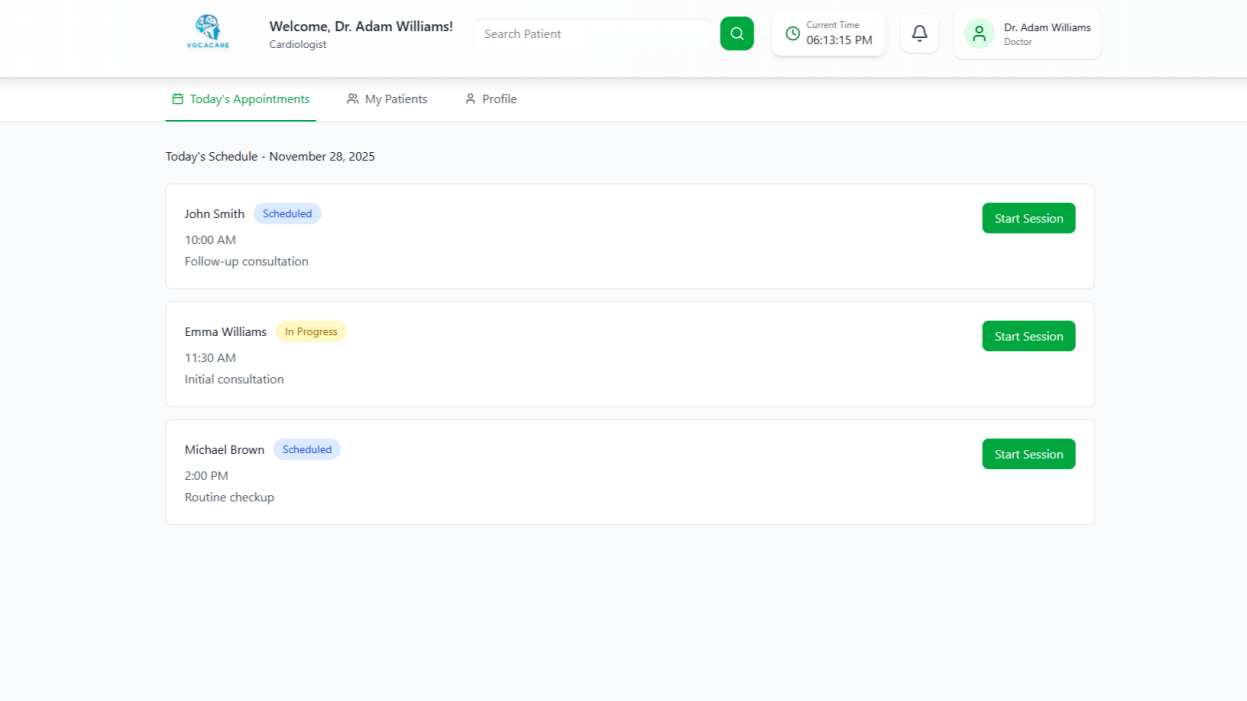
- Mark notifications as read when viewed

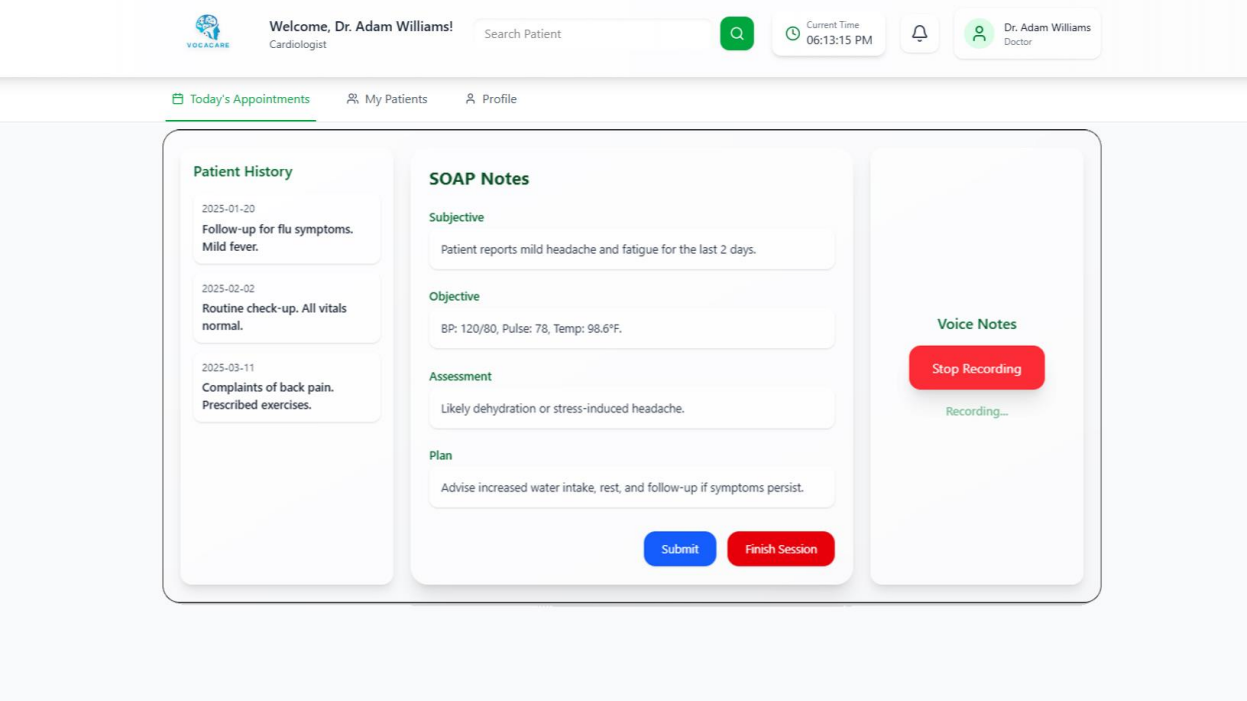
5. Clicking User Profile

System Action:

- Show menu options
- Selecting logout → return to login

5.6.5 Doctor Dashboard





Purpose & Context

Daily workflow for doctors: appointments, patients, medical history, SOAP sessions.

Element Details

Header

- Logo + welcome
- Search patients
- Current time
- Notifications
- User menu

Tabs

- Today’s Appointments
- My Patients
- Profile

Session Modal

- Left: Patient history
- Center: SOAP notes

- Right: Voice recorder
- Submit + Finish Session

User Flow

1. Doctor clicks “Start Session”

System Action:

- Opens fullscreen
- Loads patient history + SOAP template
- Initializes microphone permissions (if allowed)

2. Doctor edits SOAP Notes

System Action:

- Auto-save draft locally
- Highlight unsaved changes

3. Doctor clicks “Start Recording”

System Action:

- Ask for mic permission
- Begin audio capture
- Display “Recording…” pulse state

4. Doctor clicks “Stop Recording”

System Action:

- Save audio blob
- Display playback option (optional)

5. Doctor clicks “Submit”

System Action:

- Validate SOAP fields
- Upload notes + audio
- Close modal + mark appointment as “Completed”
- Toast: “Session Submitted.”

6. Doctor clicks “Finish Session”

System Action:

- Close modal
- Return to appointments tab

7. Doctor searches for a patient

System Action:

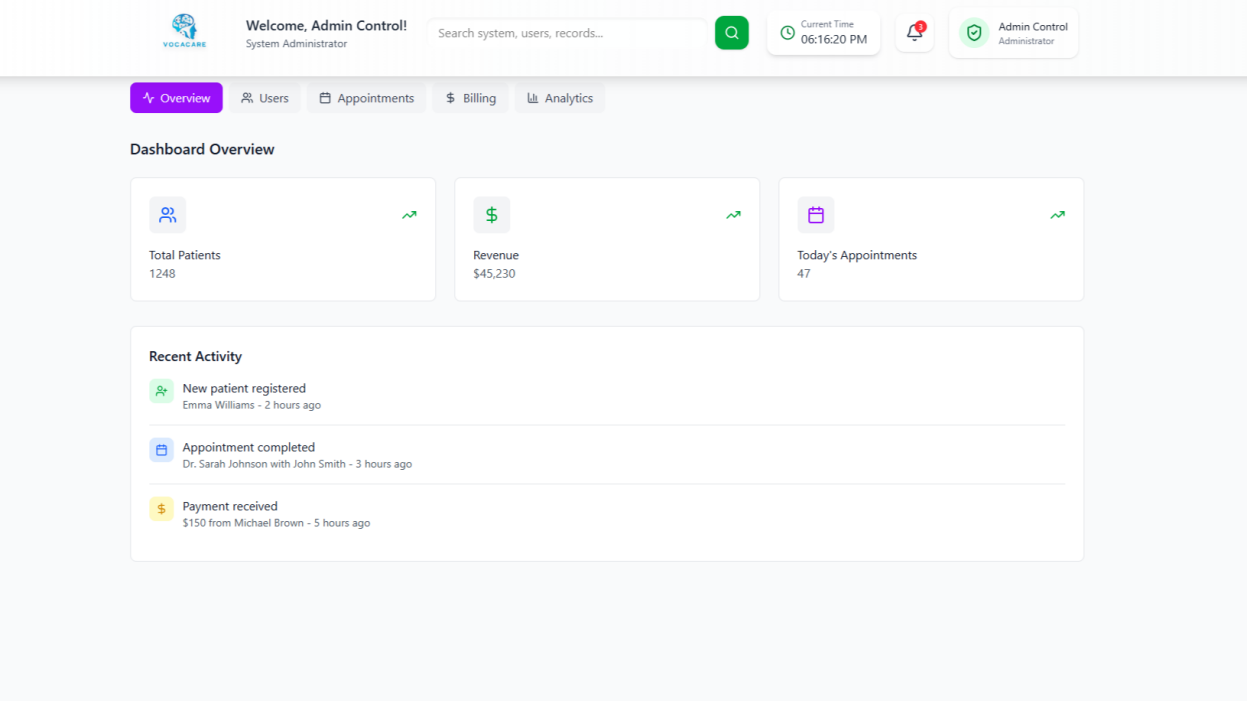
- Trigger global search
- Clicking a patient opens detailed profile

8. Editing profile

System Action:

- Save doctor info
- Refresh UI

5.6.6 Admin Dashboard



Purpose & Context

Complete management of system users, analytics, departments, billing, logs.

Element Details

Header

- Welcome admin
- Global search (search users, departments, roles)
- Current time
- Notifications
- Admin menu

Main Sections

- Overview
- Users
- Appointments
- Billing
- Analytics

User Flow

1. Admin uses global search

System Action:

- Search across:
    - users
    - doctors
    - receptionists
    - departments
    - logs
- Click result → navigate to detail page

2. Admin adds a user

System Action:

- Open create-user modal
- Fill details

- Save → User list refreshes

3. Admin views analytics

System Action:

- Load charts (appointments, revenue, user activity)
- Switch timeframe filters

4. Admin edits system roles

System Action:

- Open role editor
- Assign permissions
- Save updates

5. Notifications dropdown

System Action:

- Mark events as read

6. Logout

System Action:

- Clear session
- Redirect to login

6. References

Ref. No.	Document Title	Date of Release/ Publication	Document Source
PGBH01-2025-Proposal	Project Proposal	Sep 23, 2025	https://github.com/Hospital-Information-Management-System/Capstone/tree/main/Project%20Proposal
PGBH01-2025-FS	Functional Specification	Oct 11, 2025	https://github.com/Hospital-Information-Management-System/Capstone/tree/main/Project%20SRS

Ref. No.	Document Title	Date of Release/ Publication	Document Source
PGBH01-2025-FS	Design Document	Dec 6, 2025	https://github.com/Hospital-Information-Management-System/Capstone/tree/main/Project%20Design%20Document