DigiMed Design Specification Document

Adam Mitry, Ahmed Abdelkader, Hoda Hussein, Mohamed Khalil Brik,

Rana Taher, Yasmina Mahdy

1. Introduction

1.1 Purpose of this document

1.2 Scope of the development project

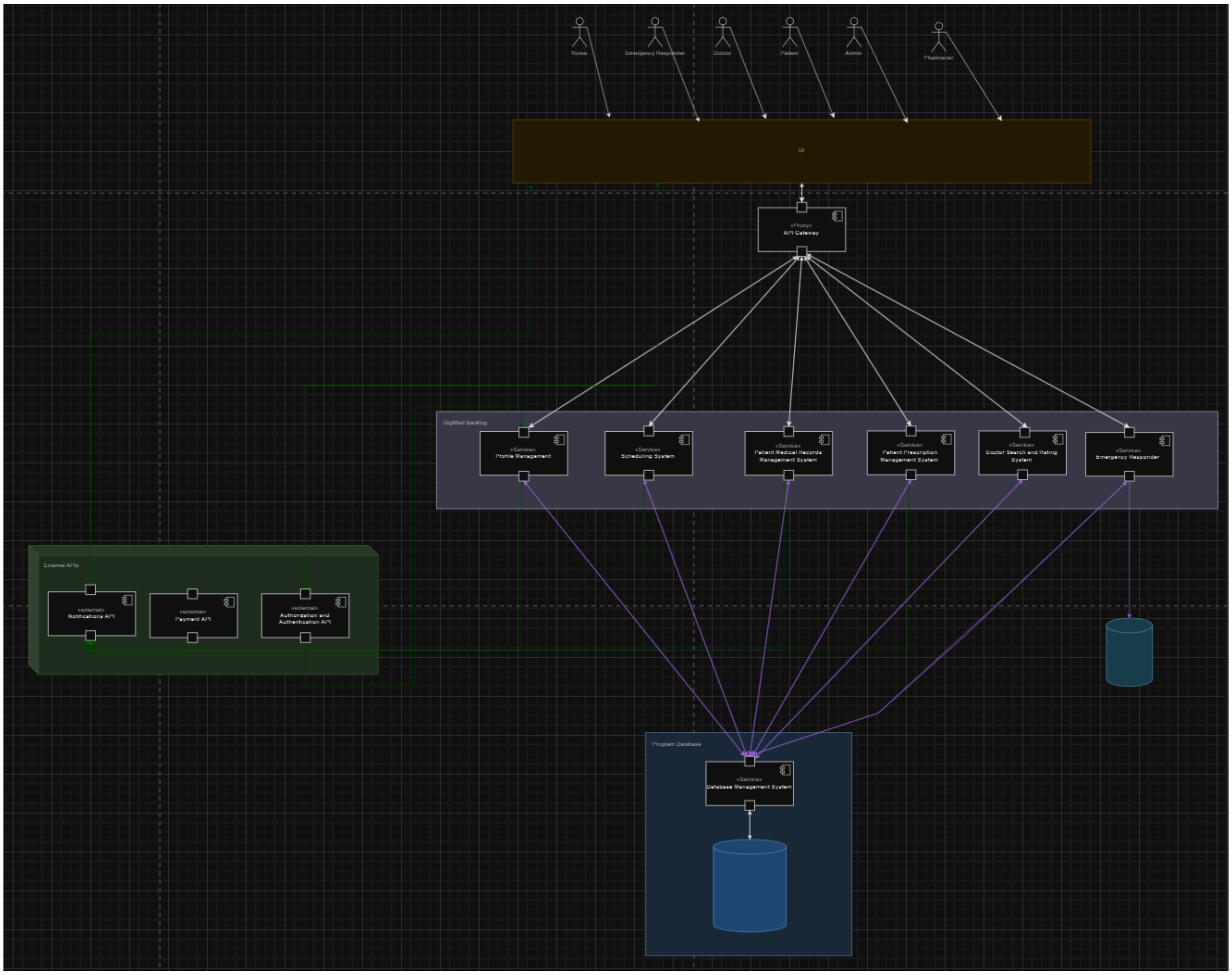
1.3 Definitions, acronyms and abbreviations

1.4 References

1.5 Overview of the document

2. System architecture description

2.1 System Architecture



Link for clearer image:
https://drive.google.com/file/d/1aoxPEG3SadUiEhjdOCFzvoNuITZHe9gD/view?usp=drive_link

2.2 Overview of components

Component	High level function
API Gateway (Reverse Proxy)	This component receives requests from the UI and forwards them to the appropriate service.
Database Management System (Service)	This component receives API calls from the Profile Management (Service), Scheduling System (Service), Patient Medical Records Management System (Service), Patient Prescription Management System (Service), Doctor Search and Rating System (Service), and Emergency Responder (Service) to control database access since not all services have access to the same data, and provides the option of scalability through load balancing and replication.
Authorization and Authentication API (External)	This component receives API calls from the UI to and calls the Profile Management (Service) to retrieve credentials for all types of profiles.
Payment API (External)	Ths component receives API calls from the Scheduling System (Service) to securely facilitate payments for appointments and calls the Notification API to send notifications to doctors and assistance of payment.
Notifications API (External)	This component receives API calls from the Scheduling System (Service), Patient Medical Records Management System (Service), Patient Prescription Management System (Service), and Payment API to send notifications to the UI.
Profile Management (Service)	This component receives API calls from the Authorization and Authentication API and API Gateway and calls the database management system to create users, change user permissions, and retrieve user information and credentials for all types of profiles.
Scheduling System (Service)	This component receives API calls from the API Gateway and calls the database management system to perform CRUD operations on a doctor's schedule, calls the Notifications API to send notifications to doctors, patients, and administrators, and calls the Payment API to

	facilitate payments for appointments.
Patient Medical Records Management System (Service)	This component receives API calls from the API Gateway and calls the database management system to perform CRUD operations on patient medical records (done by the doctor profile). It also calls the Notifications API to send notifications to patients after edits on their medical records.
Patient Prescription Management System (Service)	This component receives API calls from the API Gateway and calls the database management system to perform CRUD operations on patient prescriptions (done by the doctor profile). It also calls the Notifications API to send notifications to patients after new prescriptions are added.
Doctor Search and Rating System (Service)	This component receives API calls from the API Gateway and calls the database management system to perform CRUD operations on doctor profiles (done by patients).
Emergency Responder (Service)	This component receives API calls from the API Gateway and calls the database management system to copy specific patient medical records for its dedicated database. It performs read operations on the copied records, and can access the larger database for data if it is not available.

2.3 Structure and relationships

2.4 Requirements traceability matrix

Requirement ID	Requirement Description	Design Description Section

2.5 User Interface

3. Detailed description of components

3.1 Component template description

4. Design decisions and tradeoffs

The most important non-requirements for DigiMed are security, scalability, reliability, and maintainability. We believe that with a MicroService based architecture we can achieve that. The choice of separating the API Gateway and Authentication was to meet these requirements. While combining them simplifies the implementation, this separation allows for increased security for both, flexibility in authentication methods, easier maintainability, and specification and load management since this allows us to split client requests from authorization and authentication requests.