Assignment 2 - Project Objectives and WBS

SE 638: Software Project Management

Group #3: Shaima Albugami, Steve Greulich, Michelle Ibarra, Yiyun Zhang

October 17, 2020

Contents

[Work Summary 3](#_Toc53905210)

[1. Produce a High-Level Requirements Analysis of the IT System 4](#_Toc53905211)

[2. Analyze Component Dependencies to Define Physical Software Deliverables 6](#_Toc53905212)

[3. Produce an Architectural Model of the Software Sub-components to be Delivered 11](#_Toc53905213)

[4. Produce a Mid-level Work-Breakdown Structure Breakdown 14](#_Toc53905214)

[4.1 Delivered Components and Component Relationships 14](#_Toc53905215)

[5. Produce a System Development (Task-level) Work-Breakdown Structure 19](#_Toc53905216)

[References 25](#_Toc53905217)

### Work Summary

Each team member researched each question individually before meeting up to share their findings. After debating and gathering insight for each question, the parts were divided equally between team members. The team met twice to assign sections and perform a group review.

Shaima Albugami

* 1.2 The main functional IT system components (the logical system components) that need to be developed to support these business operations.
* 2.3 Define the high-level work-breakdown structure (WBS) for the project, based on part (a) to plan the roll-out (order and timing) of components or their subsystems for the project
* Analyze Component Dependencies to Define Physical Software Deliverables
  + Section Summary
* 4.2 Mid-level Work-Breakdown Structure breakdown
  + Phase 1
* 5.1 Produce a System Development (Task-level) Work-Breakdown Structure
  + Phase 1

Michelle Ibarra

* 1.1 Analysis of the business operations and necessary processes and sub process for the system to function.
  + Section Summary
* 2.1 Logical Software Architecture
  + Analysis of the Logical Software Architecture with respect to the 3-Phase Release
  + Created the Logical Software Architecture with respect to 3-Phase Release
* 3. System Architecture Component Analysis
  + Analysis of the four layers of the High-Level Architecture Diagram
  + Created the High-Level Architecture Diagram
* 4.1 Delivered Components and Component Relationships
  + Defining and identifying relationships (dependencies) of the delivered software components

Steven Greulich

* 2.2 Subsystem of Complex High-Risk Components in 3-Phases
  + 2.2.1 - Logical Component Dependency & Priority Analysis
  + 2.2.2 - Logical Component Roll-Out Plan
* 4.2 Mid-level Work-Breakdown Structure breakdown
  + Phase 2
* 5.1 Produce a System Development (Task-level) Work-Breakdown Structure
  + Phase 2
* 5.2 Task-level Work-Breakdown Structure Section Summary

Yiyun Zhang

* 3. System Architecture Component Analysis
  + Section Summary
* 4. Mid-level WBS Breakdown
  + Section Summary
* 4.2 Mid-level Work-Breakdown Structure breakdown
  + Phase 3
* 5.1 Produce a System Development (Task-level) Work-Breakdown Structure
  + Phase 3

### 1. Produce a High-Level Requirements Analysis of the IT System

#### The areas of business operations

This section defines the business processes and subprocess for the primary business operations.

**Phase 1**

1. Centralize and establish consistent management of Patient Healthcare Records for Doctors, Nurses, and Office Admins.
   1. *Areas of business operations*
      * Managing paper Patient Healthcare charts as well as adding and organizing exam and test records, prescriptions, and referrals.
   2. Software functions required
      * Create, view, update, and delete digital Patient Healthcare records.
      * Create, view, update, and delete Doctor exam and test orders in the digital Patient Healthcare records.
      * Create, view, update, and delete Prescriptions in the digital Patient Healthcare records.
      * Create, view, update, and delete Referrals in the digital Patient Healthcare records.
2. Create and manage Office Employee Records for Doctors, Nurses, and Office Admins.
   1. *Areas of business operations*
      * Managing paper Office Employee records, including their general profile contact information and work schedule.
   2. Software functions required
      * Create, view, update, and delete Office Employee records.
      * Create, view, update, and delete the work schedule of the Office Employees.
3. Create and manage Appointments for Doctors, Nurses, and Office Admins.
   1. *Areas of business operations*
      * Managing paper-based appointment scheduling process by the Office Employees.
   2. Software functions required
      * Create, view, update, and delete Patient Appointments.
      * View the Doctors availability (i.e. work schedule) to schedule Patient Appointments.
4. Create and manage Inventory and Ordering Supplies for Doctors, Nurses, and Office Admins.
   1. *Areas of business operations*
      * Managing paper-based inventory and ordering office equipment and supplies.
   2. Software functions required
      * Create, view, update, and delete inventory for office equipment and supplies.
      * Create, view, update, and delete orders for office equipment and supplies

**Phase 2**

1. Create and manage Prescriptions, Doctor’s Orders (exams and tests), and referral for Doctors, Nurses, and Office Admins.
   1. *Areas of business operations*
      * Managing paper-based process of writing and recording prescriptions, orders and exams, and referrals by Doctors.
   2. Software functions required
      * Create, view, and update prescriptions issued by the Doctors.
      * Create, view, and update orders and exams issued by the Doctors.
      * Create, view, and update referrals issued by the Doctors.
2. Make Patient Healthcare Records accessible to Patients.
   1. *Areas of business operations*
      * Managing paper-based appointment scheduling process by the Office Employees.
   2. Software functions required
      * View Patient Healthcare record.
      * Read and update Patient Profile record.
3. Create and manage Appointments by Patients.
   1. *Areas of business operations*
      * Managing digital appointment scheduling process by the Office Employees.
   2. Software functions required
      * Create, view, update, and delete Patient appointments.
4. Create Self-Check-in for Doctors, Nurses, Office Admins, and Patients.
   1. *Areas of business operations*
      * Managing in-person appointment check-in process by the Office Employees.
   2. Software functions required
      * Find appointment to check-in to.
      * View and update appointment check-in record.

**Phase 3**

1. Create and manage Performance Reports for Doctors and Nurses.
   1. *Areas of business operations*
      * Evaluating the healthcare treatment Doctors and Nurses provide their patients.
   2. Software functions required
      * Generate and view periodic reports on what treatments (medicine, exams, rehab, etc.) Doctors and Nurses are providing each patient.
2. Create and manage Inventory and Ordering Supplies for External Parties.
   1. *Areas of business operations*
      * Managing paper-based process of inventorying and ordering office equipment and supplies.
   2. Software functions required
      * Create, view, update and delete inventory of office equipment and supplies.
      * Create, view, update and delete order of office equipment and supplies.
3. Create and manage Billing for Nurses, Office Admins, and Patients.
   1. *Areas of business operations*
      * Managing paper-based payment and accounting process.
   2. Software functions required
      * Create, view, update and delete Patient payment records.
      * Create, view, update and delete Patient accounting records.
4. Create and manage External Services (Prescriptions, Test Orders, and Referrals) to External Parties.
   1. *Areas of business operations*
      * Managing paper-based processes for writing prescriptions to Patients and Patients filling the prescription at the Pharmacy.
      * Managing paper-based processes for writing Doctor’s Exam and Test Orders to Patients.
      * Managing paper-based processes for writing Referral to Patients.
   2. Software functions required
      * Create, view, update and delete Patient Prescriptions.
      * Create, view, update and delete Exam and Test Orders.
      * Create, view, update and delete Patient Referrals.

#### 1.2 The main functional IT system components

To support the business processes mentioned in section 1.1, the following IT system components are needed which are patient records module, employee records module, appointment module, inventory management module, report generation module, and payment and accounting module (Figure 1).

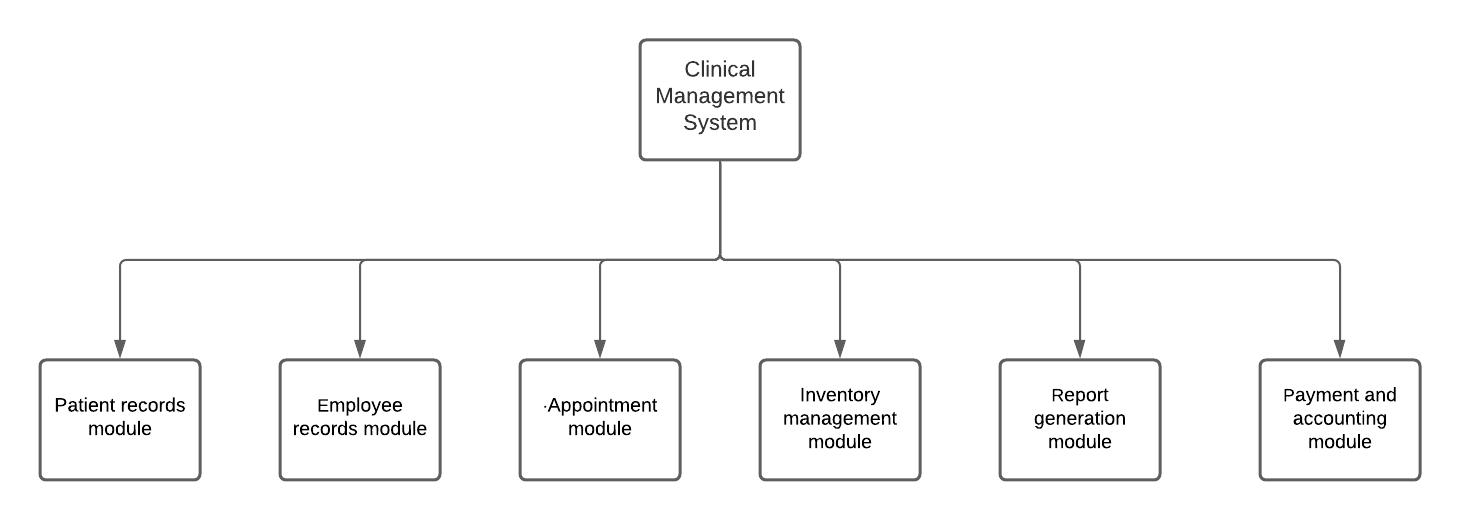


Figure 1 The main functional IT system component

#### 1. Section Summary

This section defines the primary components required for the Philadelphia Medical Group System. Each component captures what existing business area will be managed by the components. Furthermore, each component has dependencies defined to document the relationships the components have with one another. Moreover, each of the primary component lists the functions that must be made available to the respective users. As there are twelve primary components, it is essential to document the business operations affected by the development of the components and functions to aid the Business Analysts, Designers, and Development team. These definitions and the Figure 1 The main functional IT system component illustration will help team members to understand how they must plan and prioritize components included in the Philadelphia Medical Group System.

### 2. Analyze Component Dependencies to Define Physical Software Deliverables

2.1 Functional Increments of the Complex and High-Risk Components  
The Philadelphia Medical Group system requires components supporting the doctor’s office business operations and customers capabilities. This includes functions supporting Doctor, Nurses, and Office Admins day-to-day tasks such as documenting patient records, setting up appointments, checking in patients, prescribing medications, ordering exams or tests, inventorying and ordering supplies, reading Healthcare Performance Reports, and tracking office employees. The customer, the Patients, are interested in having the capability to setup appointments, check-in and view their own Patient Healthcare records. These functions and capabilities are captured as major components that are part of the Philadelphia Medical Group system are illustrated in Figure 2 Logical Software Architecture.

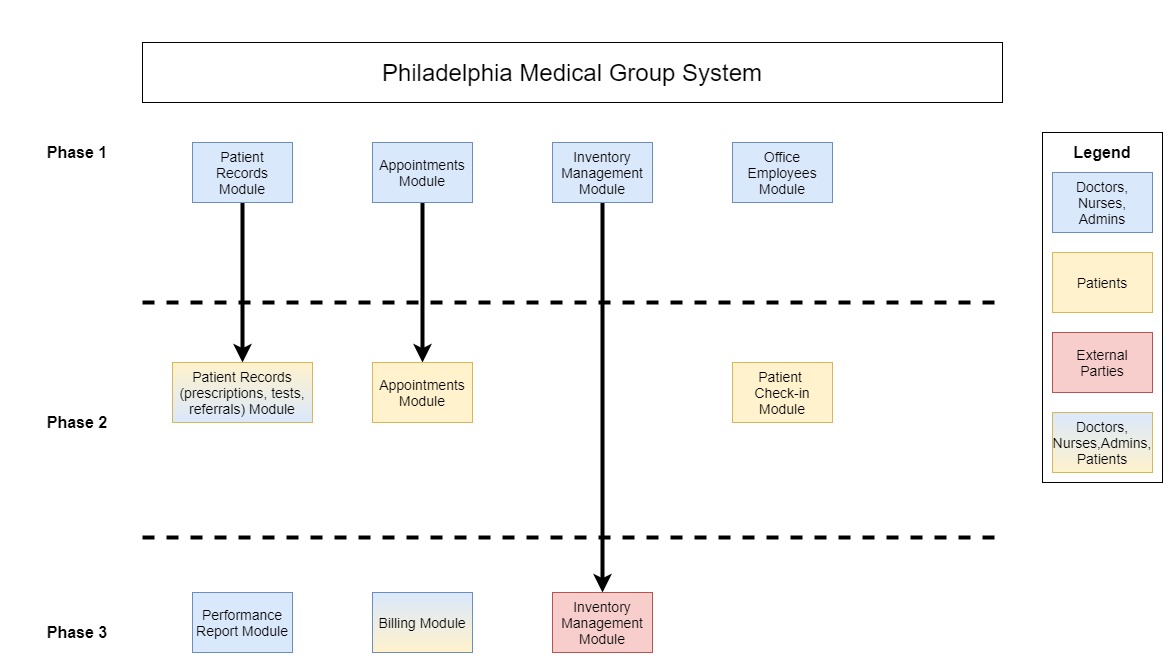


Figure 2 Logical Software Architecture

2.2 Subsystem of Complex High-Risk Components in 3-Phases

2.2.1 - Logical Component Dependency & Priority Analysis

Each component to be developed has been broken down by the following categories:

* *Dependencies* – What components need to be completed prior to development to begin. For example, component C cannot start development until Component A has been completed.
* *Priority* - How important is the development of each functionality. High would be considered for the core system. Medium would be beneficial to its users. Low would be considered nice to haves.
* *Difficulty* - How much is understood as to how to develop the functionality.
* *Risk* – How well documented and understood the requirements are. Low Risk means that the requirements are well documented and understood while high risk means that there are a lot of remaining information to be discovered

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Component | Dependency | Priority | Difficulty | Risk |
| 1. Appointment Module | - | High | Medium | Low |
| 1. Employee Records Module | - | High | Low | Low |
| 1. Patient Records | - | High | Low | Low |
| 1. Inventory Management | - | High | Low | Low |
| 1. Patient Appointment Scheduling via Portal | 1 | Medium | Medium | Low |
| 1. Patient Self Service Check-in | 1, 6 | Medium | Medium | Medium |
| 1. Patient Medical History via portal | 3 | Medium | Medium | Low |
| 1. Report Generation | All | Low | High | High |
| 1. Inventory Ordering | 4 | Low | Medium | Low |
| 1. Payment And Accounting Module | - | Low | Low | Medium |
| 1. Prescription Ordering | - | Low | Medium | High |
| 1. Referral Processing | - | Low | Medium | High |

2.2.2 - Logical Component Roll-Out Plan

**Phase 1 - Core System used by the Doctors, Nurses, and Office Admin**

* Appointment module (Appointment Scheduling, Appointment History, Virtual Appointments)
* Employee records module (Employee Scheduling and payroll)
* Patient records module (Medical Records only)
* Inventory management module (internal inventory).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Baseline** – Features above this line are deemed core system functionalities

**Phase 2 - Patient Portal**

* Patient records module (Prescriptions, Test Results, Referrals)
* Patient Appointment module (self service)
* Patient Self Service Check-in

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Patient Facing** – Features above this line will be utilized by the patients.

**Phase 3 - Referrals, Prescription, Ordering supplies, Reports**

* Report generation module
* Inventory management module (external ordering)
* Payment and accounting module (invoicing, payment history, insurance claims)
* Outgoing/external sending of prescriptions, test results, and referrals to other party (e.g. Pharmacy, Lab, etc.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**External Interfaces** – Features above this line require previous components to be in place prior to development.

**Note**: In assignment 1, we stated that the priority in system implementation would be for the appointment scheduling system. While we still aim to prioritize appointment scheduling in Phase 1 (for use by the staff), we are expanding Phase 1 to develop and release more modules. Upon reevaluation, we prioritized the components needed for the core system to be successfully adopted by its core users.

#### 2.3 High-Level Work-Breakdown Structure

The high-level work-breakdown structure below Figure 3 High-Level Work-Breakdown Structure shows the components of the system arranged in three different phases. Phase 1 is the core components of the system; Phase 2 is for the components that would help the clinic to be more efficient. For example: having the patient self-check-in to speed the check-in process. Finally, Phase 3 is for the components that will help the clinic staff better manage their performance in order to ensure consistency in how Patient receive the best healthcare.

Some of the modules are going to take more than one phase as they consist of many sub-functionalities. These sub-functionalities are prioritized in phases depending on whether they are an integral part of the core system and crucial for the clinic. For example, the patient record module has many sub-functionalities, such as patient medical records and referrals. The patient medical record will be developed in Phase 1; however, the referrals will be added in the Phase 2.

Having the system developed in modules helps us be focused on developing the core components while tackling all the issues that might come up. It is better than developing the system all at once and then getting feedback that might take longer to fix and affect functionalities dependent on each other.

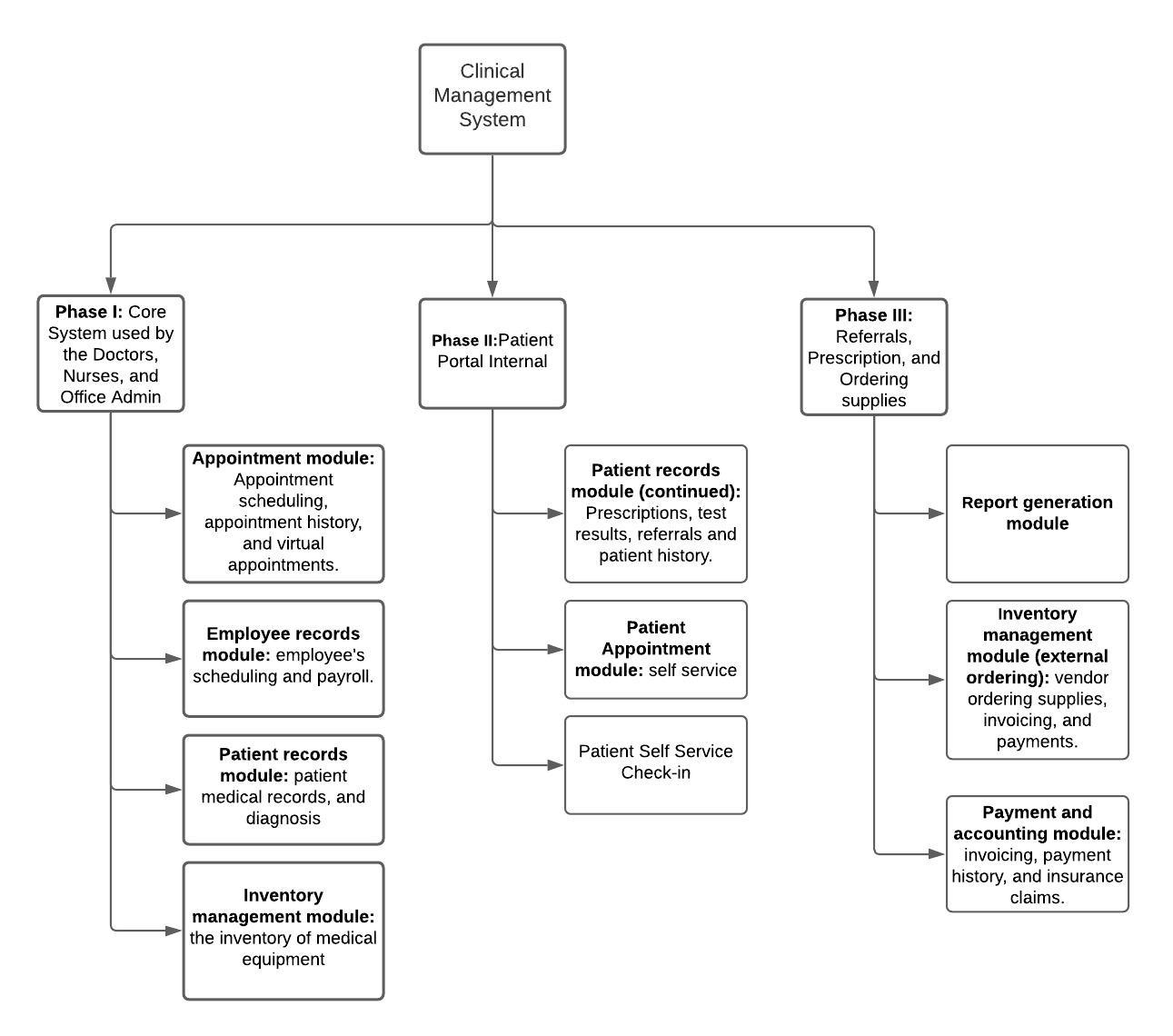


Figure 3 High-Level Work-Breakdown Structure

#### 2. Section Summary

In this section, we presented the logical software architecture of The Philadelphia Medical Group system. The diagrams showed the different components, the phase they are going to be implemented, and the users of these components. Some of these components are implemented incrementally throughout several phases. The incremental implementation ensures that the core components get released first. Then, we presented the dependency between these components, priority, difficulty, and risk. Based on that, we developed a rollout plan for the logical components. The plan is divided into three phases in which each focused on developing certain modules. Moreover, the high-level work-breakdown structure was shown to describe the expected outcome of each phase. Phase 1 has the core system used by the Doctors, Nurses, and Office Admin, which are the appointment module, employee module, patient records, module, and inventory management module; Phase 2 has the patient portal internal; and Phase 3 has report module, external inventory ordering, and payment module.

### 3. Produce an Architectural Model of the Software Sub-components to be Delivered

In this section introduces the high-level architecture of the Philadelphia Medical Group system. A high-level architecture serves to illustrate the system components and how they interface with one another. The Philadelphia Medical Group system is adopting a microservice architecture utilizing REST APIs. Microsoft defines microservice architecture as “a collection of small, autonomous service. Each service is self-contained and should implement a single business capability,” (Microsoft). Also, REST API is an acronym for Representational State Transfer Application Programming Interface. The advantage of this microservice architecture is the services are loosely coupled and are designed for the system to have less dependencies on data retrieval. This means services can be deployed, rebuilt, and redeployed without disrupting the entire system.

In the high-level architecture diagram Figure 4 High-Level Architecture of Software Components, the four areas of the diagram include the User Interface, API Gateway, RESTful API Service Server, and Databases. The API Gateway, RESTful API Service Server and Database are part of the underlying Infrastructure. This system focus’ on three persona groups: Doctors and Nurses, Office Admins, and Patients. Each user interface is designed for creating, viewing, editing, submitting data via the modules into the Philadelphia Medical group system. Access to modules depends on the persona role. The Philadelphia Medical Group system is accessible through a web-based application for all personas.For the Doctors and Nurses, the primary user interfaces include Patient Record, Appointment, Supply Inventory, Office Employees, and Performance Report modules. For Office Admins, the primary user interfaces include Appointment, Check-in, Supply Inventory, Office Employees, and Billing modules. The Patients primary user interfaces includes the Patient Records, Appointment, Check-in, and Billing modules.

The API Gateway layer is between the User Interface layer and RESTful APIs. The purpose of the API Gateway is to serve as the central point for all API calls. The API Gateway manages the API calls by proxying the API calls to the appropriate API service. API Gateways provide load balancing service, rate limiting, monitoring, security, and other capabilities for analyzing the performance of your API services. Load balancing capability allows for balance of incoming data calls to various node to increase performance of an API endpoint (HAProxy). Rate limiting is another feature that allows API Gateway Admins to limit the number of API calls a client can sent to the API services within a specified period (API.DATA.GOV). Rate limiting helps prevent abuse in client submitting excessive API calls, particularly when the API services are accessible to external consumers (Pharmacy and Equipment Suppliers). The API Gateway includes all the API calls for Performance Report, Patient Record, Supply Inventory and Ordering, Office Employees, Appointment, Check-in, and Billing API services.

The RESTful APIs are providing a standard method for exchanging data. OpenAPI Specification is a standard interface for developing, consuming, and visualizing REST APIs (Swagger). Using standard interface for REST APIS allow for better communication with other external services that do not have common technology stacks, allowing for more interoperability between systems. Interoperability will be particularly essential when the Philadelphia Medical Group system integrates with external API services such as those provided by the Pharmacy and Equipment Supplier. For the scope of this project, the API services create will include Patient Profile, Healthcare Records, Exam & Test Orders, Prescription, Referral, Supply Inventory, Supply Ordering, Employee Profile, Schedule, Appointment, Check-in, Payment, and Accounting API services.

#### High-Level System Architecture

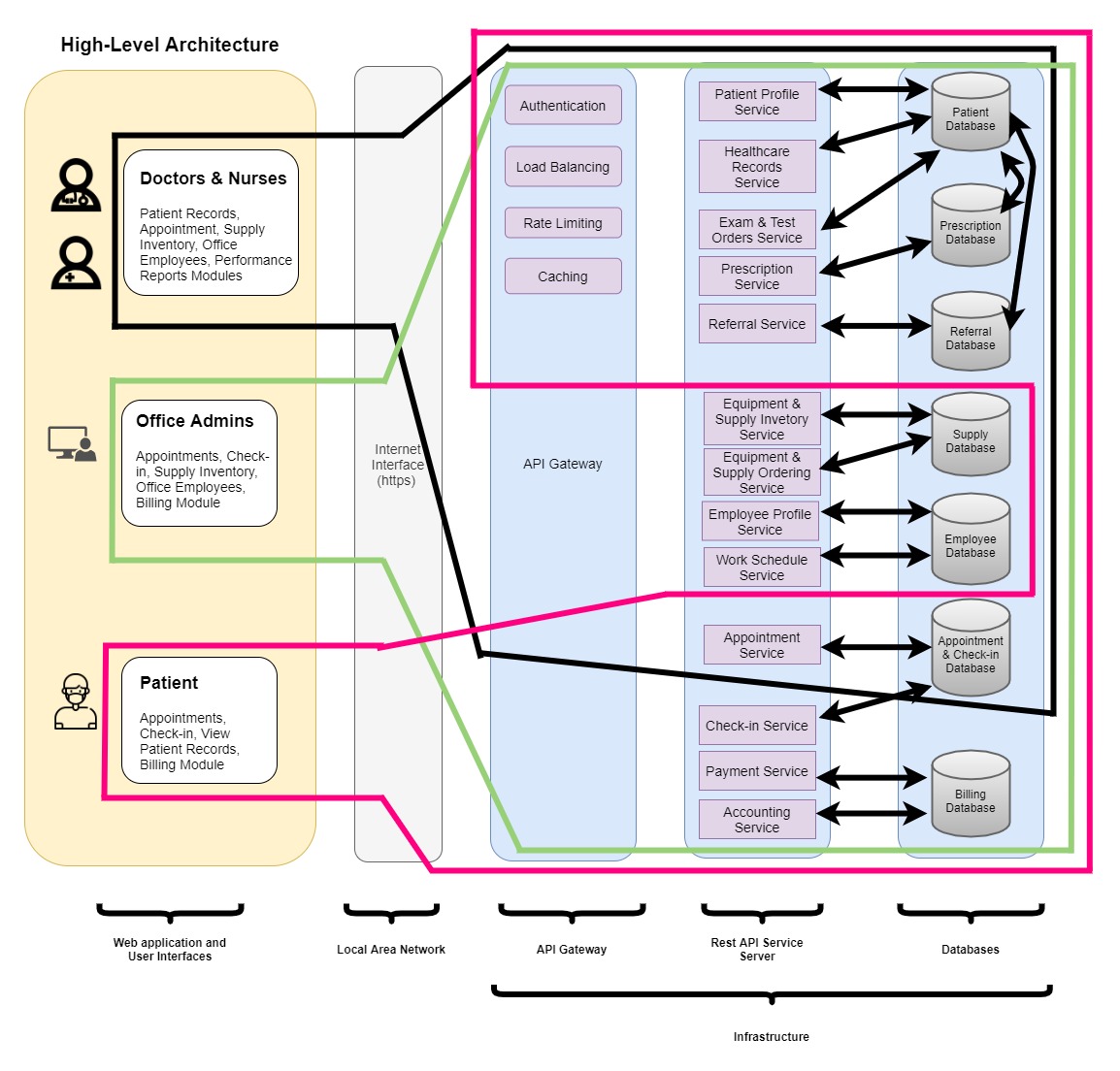


Figure 4 High-Level Architecture of Software Components

#### 3. Section Summary

By defining this architecture model, we can list all modules and sub-components need to be deployed to fulfill our proposed software system. There will be five major components, for each component, there will be several sub-components that will perform functional behaviors. The three major user groups of the system are: Doctors, Nurses, Office Admins and Patients. Each of them will have different authorities to access the various parts of the system based on their role and responsibilities. In local area network, the internet interface will be the portal for different user groups to access the system. During the data exchange process, the API Gateway will manage requests and user traffic, including authentication, load balancing, rate limiting and caching. For data requests, there will 13 different types of data, corresponding to 7 databases. The databases store all data that required by the system.

Furthermore, we also defined detailed, specific technology stacks that will be used for our proposed software system. For the overall architecture, the RESTful APIs model is a popular, well-developed and reliable software architecture. By using it, we can deploy our system in a standardized, effective and supportive manner. We also mentioned API Umbrella as the API Gateway service and OpenAPI specification as the Swagger specification. These technology stacks provide us working solutions to implement those sub-components.

### 4. Produce a Mid-level Work-Breakdown Structure Breakdown

### 4.1 Delivered Components and Component Relationships

**User Interface (UI)**

Each user interface is intended to serve each user’s primary needs.

1. Doctors and Nurses UI
   1. Description: this UI makes the Patient Healthcare Records, Appointment, Supply Inventory & Ordering, Performance Report and Office Employee modules available.
   2. Dependencies: this UI depends on the internet for connectivity and API Gateway for routing request from client web-application to the API services.
2. Office Admins UI
   1. Description: this UI makes the Patient Healthcare Records, Appointment, Supply Inventory & Ordering, Office Employee, Check-in, and Billing modules available.
   2. Dependencies: this UI depends on the internet for connectivity and API Gateway for routing request from client web-application to the API services.
3. Patient UI
   1. Description: this UI makes the Patient Healthcare Records, Appointment, and Check-in, modules available.
   2. Dependencies: this UI depends on the internet for connectivity and API Gateway for routing request from client web-application to the API services.

**API Gateway**

1. Authentication
   1. Description: authenticate users (Doctors, Nurses, Office Admins, and Patients) to access certain API services for retrieving data.
   2. Dependencies: requires Developer to use HTTP Methods and logic to authorize access to specific users (Kong).
2. Load Balancing
   1. Description: manage API requests to API endpoints, balance load to different nodes as need.
   2. Dependencies: requires Developer to make multiple nodes available for the API Gateway to point to.
3. Rate Limiting
   1. Description: limit API calls during a defined period (e.g. 1,000 API calls per minute).
   2. Dependencies: User inputting the definition of the rate limit.
4. Caching
   1. Description: store previously searched and returned API calls to improve API performance and reduce traffic to the APIs (AWS).
   2. Dependencies: Developer add API caching and specifies the size of the cache (e.g. Gigabytes) (AWS).

**API Services**

1. Patient Profile Service
   1. Description: sends HTTP methods (e.g. GET, POST, PUT, DELETE) and return Patient Profile data.
   2. Dependencies: this API service is dependent on the API Gateway for proxying the HTTP methods and the Patient Database for returning data from the API call.
2. Healthcare Records Service
   1. Description: sends HTTP methods (e.g. GET, POST, PUT, DELETE) and return Patient Healthcare Record data.
   2. Dependencies: this API service is dependent on the API Gateway for proxying the HTTP methods and the Patient Database for returning data from the API call.
3. Exam & Test Orders Service
   1. Description: sends HTTP methods (e.g. GET, POST, PUT, DELETE) and return Patient Exam and Test Order data.
   2. Dependencies: this API service is dependent on the API Gateway for proxying the HTTP methods and the Patient Database for returning data from the API call.
4. Prescription Service
   1. Description: sends HTTP methods (e.g. GET, POST, PUT, DELETE) and return Prescription data.
   2. Dependencies: this API service is dependent on the API Gateway for proxying the HTTP methods and the Prescription Database for returning data from the API call.
5. Referral Service
   1. Description: sends HTTP methods (e.g. GET, POST, PUT, DELETE) and return Referral data.
   2. Dependencies: this API service is dependent on the API Gateway for proxying the HTTP methods and the Referral Database for returning data from the API call.
6. Inventory Service
   1. Description: sends HTTP methods (e.g. GET, POST, PUT, DELETE) and return Supply Inventory data.
   2. Dependencies: this API service is dependent on the API Gateway for proxying the HTTP methods and the Supply Database for returning data from the API call.
7. Ordering Service
   1. Description: sends HTTP methods (e.g. GET, POST, PUT, DELETE) and return Supply Order data.
   2. Dependencies: this API service is dependent on the API Gateway for proxying the HTTP methods and the Supply Database for returning data from the API call.
8. Appointment Service
   1. Description: sends HTTP methods (e.g. GET, POST, PUT, DELETE) and return Patient Appointment data.
   2. Dependencies: this API service is dependent on the API Gateway for proxying the HTTP methods and the Appointment & Check-in Database for returning data from the API call.
9. Check-in Service
   1. Description: sends HTTP methods (e.g. GET, POST, PUT, DELETE) and return Patient Check-in data.
   2. Dependencies: this API service is dependent on the API Gateway for proxying the HTTP methods and the Appointment & Check-in Database for returning data from the API call.
10. Payment Service
    1. Description: sends HTTP methods (e.g. GET, POST, PUT, DELETE) and return Patient Payment data.
    2. Dependencies: this API service is dependent on the API Gateway for proxying the HTTP methods and the Billing Database for returning data from the API call.
11. Accounting Service
    1. Description: sends HTTP methods (e.g. GET, POST, PUT, DELETE) and return Accounting data.
    2. Dependencies: this API service is dependent on the API Gateway for proxying the HTTP methods and the Billing Database for returning data from the API call.

**Databases**

1. Patient Database
   1. Description: stores the Patient Profile and Healthcare records. This includes prescriptions, exam and test orders, and referral records.
   2. Dependencies: this database is dependent on the following API services to send HTTP methods (e.g. GET, POST, PUT, DELETE) and return data: Patient Profile Service, Healthcare Record Service, and Exam and Test Orders Service. This database also has relationships with the Prescription and Referral Database.
2. Prescription Database
   1. Description: stores the list of available Prescription Medicines for Doctors to prescribe to their patients.
   2. Dependencies: this database is dependent on the Prescription API service to send HTTP methods (e.g. GET, POST, PUT, DELETE) and return data.
3. Referral Database
   1. Description: stores the list of available Referral options Doctors have for their Patients.
   2. Dependencies: this database is dependent on the Referral API service to send HTTP methods (e.g. GET, POST, PUT, DELETE) and return data.
4. Supply Database
   1. Description: stores the list of available equipment and supplies as well as the orders for equipment and supplies.
   2. Dependencies: this database is dependent on the Supply Inventory Service and Supply Ordering Service API service to send HTTP methods (e.g. GET, POST, PUT, DELETE) and return data.
5. Employee Database
   1. Description: stores the list of Office Employees, including the Doctors, Nurses and Office Admins. This database also stores the work schedules of the Office Employees.
   2. Dependencies: this database is dependent on the Employee Profile Service and Work Schedule Service API service to send HTTP methods (e.g. GET, POST, PUT, DELETE) and return data.
6. Appointment and Check-in Database
   1. Description: stores the Patient appointment and check-in information.
   2. Dependencies: this database is dependent on the Appointment Service and Check-in Service API service to send HTTP methods (e.g. GET, POST, PUT, DELETE) and return data.
7. Billing Database
   1. Description: stores the Billing data, specifically the Patient payment and accounting information.
   2. Dependencies: this database is dependent on the Payment Service and Accounting Service API service to send HTTP methods (e.g. GET, POST, PUT, DELETE) and return data.

#### 4.2 Mid-level Work-Breakdown Structure breakdown

**Phase 1: Core System used by the Doctors, Nurses, and Office Admin:**

* *Develop Appointment module:*
  + Client App: Appointment scheduling, history and virtual appointments application
  + REST API Service Server: Appointment management
  + DBMS: Install & Configure Appointment DBMS & SQL Interface
* *Develop employee records module:*
  + Client App: Employee's assigning and payroll application
  + REST API Service Server: Employee's assigning and payroll management
  + DBMS: Install & Configure Employee's assigning and payroll DBMS & SQL Interface
* *Develop Patient records module:*
  + Client App: Patient Medical Records Diagnosis Application
  + REST API Service Server: Patient Medical Records Diagnosis Management
  + DBMS: Install & Configure Patient Medical Records Diagnosis DBMS & SQL Interface
* *Develop Inventory Management Module:*
  + Client App: The Inventory of Medical Equipment Application
  + REST API Service Server: The Inventory of Medical Equipment Management
  + DBMS: Install & Configure the Inventory of Medical Equipment DBMS & SQL Interface

**Phase 2: Patient Portal:**

* *Develop Patient Facing Record History*
  + Client App: Patient Portal - Medical Records Application
  + REST API Service Server: Patient Medical Records Diagnosis Management (enhanced to allow for patient roles to access)
  + DBMS: Configure Patient Medical Records Diagnosis DBMS & SQL Interface for patient queries
* *Develop Patient Facing Appointment Module*
  + Client App: Patient Portal – Appointment Scheduling
  + REST API Service Server: Appointment Management
  + DBMS: Configure Appointment DBMS& SQL Interface to allow for patient check-ins
* *Develop Patient Self Service Check-in*
  + Client App: Patient Portal – Self Service Check-in
  + REST API Service Server: Appointment Check-in
  + DBMS: Install & configure Appointment Check-in DBMS & SQL

**Phase 3: Referrals, Prescription, Ordering supplies, Reports:**

* *Develop report generation module*
* Client App: Report management application
* REST API Service Server: Performance report management
* DBMS: Configure performance report based on existing information, DBMS & SQL Interface to allow report generation.
* *Develop external inventory management module*
  + Client App: External portal - The inventory of medical equipment application
  + REST API Service Server: The inventory of medical equipment management for external order
  + DBMS: Install & configure the inventory of medical equipment DBMS & SQL Interface
* *Develop payment and accounting module*
  + Client App: The payment and accounting application
  + REST API Service Server: The payment and accounting management
  + DBMS: Install & configure the payment and accounting DBMS & SQL Interface
* *Develop outgoing module*
  + Client App: The outgoing business application
  + REST API Service Server: The outgoing business management
  + DBMS: Install & configure the outgoing business DBMS & SQL Interface to allow sending prescriptions, test results and referrals to third parties.

#### 4. Section Summary

The purpose of this section is for us to think about all components and sub-components need to be developed, reorganize and reschedule each of them based on our priorities. For the four major components: User Interface, API Gateway, API Services and Databases, their sub-components are the executors of the business workflow. Each sub-component performs unique functional behavior, as the description of the sub-component explains. The sub-components have unique dependencies to other sub-components under the same major component, they don’t have any necessary relationship to other major components or their sub-components due to the architecture performance requirement: high cohesion and low coupling.

Based on our priority analysis, we conclude 3 phases of our development plan, including dividing similar or the same module into different phases to fulfill our priority requirements. Such as develop Patient record module for diagnosis in Phase 1 and leave the other records in Phase 3 and develop inventory management module for internal in Phase 1 and develop the same module for external in Phase 3. For all planned module, there will be a client application facing to different user group, a REST API Service Server performs management and a DBMS that require installation and configuration of the interface.

### 5. Produce a System Development (Task-level) Work-Breakdown Structure

**Phase 1**

**Develop Appointment module**

DBMS:

* Install & Configure DBMS Engine & SQL Interface for Appointment
* Test SQL interface software

REST API Service Server:

* Requirements gathering
* Requirements documentation
* Software design
* Software code
* Software test

Client App:

* Requirements gathering
* Requirements documentation
* User Interface design
* Software design
* Software code
* Software test

**Develop employee records module**

DBMS:

* Install & Configure DBMS Engine & SQL Interface for employee records
* Test SQL interface software

REST API Service Server:

* Requirements gathering
* Requirements documentation
* Software design
* Software code
* Software test

Client App:

* Requirements gathering
* Requirements documentation
* Software design
* User Interface design
* Software code
* Software test

**Develop Patient records module**

DBMS:

* Install & Configure DBMS Engine & SQL Interface for Patient records
* Test SQL interface software

REST API Service Server:

* Requirements gathering
* Requirements documentation
* Software design
* Software code
* Software test

Client App:

* Requirements gathering
* Requirements documentation
* Software design
* User Interface design
* Software code
* Software test

**Develop Inventory Management Module**

DBMS:

* Install & Configure DBMS Engine & SQL Interface for Inventory Management
* Test SQL interface software

REST API Service Server:

* Requirements gathering
* Requirements documentation
* Software design
* Software code
* Software test

Client App:

* Requirements gathering
* Requirements documentation
* Software design
* User Interface design
* Software code
* Software test

Phase 1 Activities

* Integration testing of all Phase 1 functionalities

**Phase 2**

**Develop Patient Facing Record History**

DBMS:

* Configure DBMS Engine & SQL Interface for patient history (patient portal, referrals, prescriptions, test results)
* Test SQL interface software

REST API Service Server:

* Requirements gathering
* Requirements documentation
* Software design
* Software code
* Software test

Client App:

* Requirements gathering
* Requirements documentation
* User Interface design
* Software design
* Software code
* Software test

**Develop Patient Facing Appointment Module**

DBMS:

* Configure DBMS Engine & SQL Interface for patient appointment scheduling
* Test SQL interface software

REST API Service Server:

* Requirements gathering
* Requirements documentation
* Software design (re-design as needed)
* Software code
* Test of new features

Client App:

* Requirements gathering
* Requirements documentation
* Software design (re-design as needed)
* User Interface design
* Test of new features

**Develop Patient Self Service Check-in**

DBMS:

* Install & Configure DBMS Engine & SQL Interface
* Test SQL interface software

REST API Service Server:

* Requirements gathering
* Requirements documentation
* Software design
* Software code
* Software test

Client App:

* Requirements gathering
* Requirements documentation
* Software design
* User Interface design
* Software code
* Software test

Phase 2 Activities

* Integration testing of all Phase 2 functionalities
* Regression testing of all Phase 1 functionalities

**Phase 3**

**Develop report generation module**

DBMS:

* Install & Configure DBMS Engine & SQL Interface
* Test SQL interface software

REST API Service Server:

* Requirements gathering
* Requirements documentation
* Software design
* Software code
* Software test

Client App:

* Requirements gathering
* Requirements documentation
* User Interface design
* Software design
* Software code
* Software test

**Develop external inventory management module**

DBMS:

* Install & Configure DBMS Engine & SQL Interface
* Test SQL interface software

REST API Service Server:

* Requirements gathering
* Requirements documentation
* Software design (re-design as needed)
* Software code
* Software test

Client App:

* Requirements gathering
* Requirements documentation
* User Interface design
* Software design
* Software code
* Software test

**Develop payment and accounting module**

DBMS:

* Install & Configure DBMS Engine & SQL Interface
* Test SQL interface software

REST API Service Server:

* Requirements gathering
* Requirements documentation
* Software design
* Software code
* Software test

Client App:

* Requirements gathering
* Requirements documentation
* User Interface design
* Software design
* Software code
* Software test

**Develop outgoing module**

DBMS:

* Install & Configure DBMS Engine & SQL Interface
* Test SQL interface software

REST API Service Server:

* Requirements gathering
* Requirements documentation
* Software design
* Software code
* Software test

Client App:

* Requirements gathering
* Requirements documentation
* User Interface design
* Software design
* Software code
* Software test

Phase 3 Activities

* Integration testing of all Phase 3 functionalities
* Regression testing of all Phase 1, 2 & 3 functionalities
* User Acceptance Testing of all Phase 1, 2 & 3 functionalities
* Performance Testing of all Phase 1, 2 & 3 functionalities

#### 5. Section Summary

The way that the phases were laid out was to setup the core platform for the Philadelphia Medical Group’s medical staff first. This initial core system would be the backbone for all the subsequent phases (and perhaps future enhancements as well) to be built upon.

In some instances, during Phase 2 and Phase 3, we leverage systems setup in previous phases and enhance it further. For example, the scheduling and medical history in Phase 2 was able to leverage the existing setup from Phase 1, so only configuring was needed for the database and setting up of the web server.

Throughout every phase, there will be several rounds of testing. While each component is being developed or enhanced, there will be unit testing of each functionality within the system. As the development for the phase ends, a round of integration testing will occur for the newly developed features to each that there is consistency and flow when navigating within the system. Finally, there will be a round of regression testing of the existing functionality to ensure that nothing was broken in the current phase’s development.

### References

1. AWS. Amazon API Gateway FAQs. Amazon API Gateway. <https://aws.amazon.com/api-gateway/faqs/> .
2. Data.gov. About api.data.gov. API.DATA.GOV. <https://api.data.gov/about/>.
3. HAProxy. (2019, October 01). Using HAProxy as an API Gateway, Part 1 [Introduction]. Blog. <https://www.haproxy.com/blog/using-haproxy-as-an-api-gateway-part-1/>.
4. Kong. Microservices Authentication and Authorization using an API Gateway. API Gateway Authentication. <https://konghq.com/learning-center/api-gateway/api-gateway-authentication/> .
5. Microsoft. (2019, October 30). *Microservices architecture style.* Microsoft Docs. <https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices>.
6. Swagger. (2020, February 02). Specification. OpenAPI Specification. <https://swagger.io/specification/>.