deti universidade de aveiro departamento de eletrónica, telecomunicações e informática

TQS: Product specification report

Gonçalo Lopes [107572], Rodrigo Graça [107634], Tiago Cruz [108615]

[This report should be written as the main source of technical documentation on the project, clarifying the functional scope and architectural choices. Provide concise, but informative content, allowing other software engineers to understand the product and quickly access the related resources.
Tips on the expected content, along the document, are meant to be removed.
You may use English or Portuguese; do not mix.]

1 Introduction

1.1 Overview of the project

This project, designed as part of the Total Quality Software (TQS) course, aims to develop a digital solution for the New Private Health Initiative (newPhi), a hypothetical healthcare system. The primary objective is to design and implement an integrated IT solution that streamlines patient appointments and admissions while excluding clinical records for now. This IT solution enhances patient and staff experience by providing efficient, robust, and user-friendly digital services. It is developed to handle pre-appointment scheduling, reception services, and post-encounter processes like billing and presence confirmation, focusing on offering competitive services by improving accessibility and operational efficiency in healthcare settings.

1.2 Limitations

While newPhi aims to be comprehensive, certain limitations are inherent in the current scope of the project:

Clinical Records Integration: At this stage, integration with existing clinical records systems is not included. This functionality is essential for a complete healthcare management system and is planned for future phases.
Mobile Application: The mobile version of the Patient portal, which would allow for greater accessibility and convenience, is also planned but not yet implemented.
Advanced Analytics: Use of data analytics for optimizing hospital operations and patient flow is considered but currently outside the project scope.

2 Product concept and requirements

2.1 Vision statement

newPhi is designed to revolutionize the way patients interact with healthcare facilities by providing a streamlined, intuitive digital platform for scheduling and managing healthcare appointments. This system addresses the high-level business problem of inefficient healthcare management, where patients struggle with long wait times, complex booking procedures, and inefficient administrative processes. By consolidating patient management, administrative tasks, and digital display systems into a unified platform, newPhi ensures a smoother operational flow and enhanced patient care experience

2.2 Personas and scenarios

Commented [IO1]: or, in alternative, actors and use cases

Personas

- **Persona 1: Emma (Patient)**
  - **Demographics:** Age 32, employed, tech-savvy.
  - **Behaviors:** Prefers online scheduling, uses mobile apps for health management.
  - **Goals:** Quickly find and book appointments, easy check-in, minimal waiting times.
- **Persona 2: Dr. Harris (Healthcare Provider)**
  - **Demographics:** Age 45, general practitioner.
  - **Behaviors:** Needs efficient tools to manage patient flow.
  - **Goals:** Reduce administrative tasks, seamless access to patient appointment schedules.
- **Persona 3: Linda (Reception Staff)**
  - **Demographics:** Age 38, familiar with basic IT tools.
  - **Behaviors:** Manages patient registration and billing.
  - **Goals:** Streamline patient check-ins, efficient handling of payments and inquiries.

Main Scenarios

Here are the key scenarios that the system will need to support:

- **Scenario 1: Booking an Appointment (Emma)**
  - Emma logs into *Patient*, searches for a dermatologist at the nearest hospital, views available slots, and books an appointment.
- **Scenario 2: Patient Check-in (Linda)**
  - Linda uses *Desk* to check in arriving patients, verifies their details, and updates their status in the system.
- **Scenario 3: Calling Patients to Consultation (Dr. Harris)**
  - Dr. Harris uses *Desk* to view his daily schedule and *Boards* to call patients from the waiting room based on their appointment times and priority status.

2.3 Project epics and priorities

The development process is organized into three primary epics, each encompassing a broad set of functionalities critical to the overall system. The implementation is planned incrementally over several iterations or releases, focusing on delivering essential features early and refining or expanding the capabilities in subsequent phases. Below is an indicative plan for the incremental implementation of these epics, highlighting the functionalities to be achieved in each.

Epic 1: Patient Management

Objective: Enhance patient experience by providing efficient tools for managing their healthcare interactions.

Priorities and Incremental Implementation:

Phase 1 (Initial Release):
Develop basic patient profile management, including registration and data update functionalities.
Implement core appointment booking functionality with basic search filters (e.g., by specialty, date).

Phase 2:
Enhance the appointment booking system with additional filters (e.g., location, doctor ratings).
Introduce appointment rescheduling and cancellation capabilities.

Phase 3:
Enable access to and management of personal care files, allowing patients to view their medical history and past appointment details.
Integrate patient feedback mechanisms directly into the care file access features.

Epic 2: Administrative Interface

Objective: Streamline administrative processes to improve efficiency at the reception desk and reduce patient wait times.

Priorities and Incremental Implementation:

Phase 1 (Initial Release):
Set up basic patient registration and check-in at the reception.
Develop a simplistic billing module for processing payments.

Phase 2:
Enhance the administrative dashboard with real-time queue management tools.
Implement advanced features in the billing module, including support for multiple payment methods and insurance claims.

Phase 3:
Integrate administrative tools with the patient management system for seamless data flow and improved data accuracy.
Add reporting capabilities for administrative tasks to aid in decision-making and operational efficiency.

Epic 3: Information Display System

Objective: Provide real-time information to both staff and patients through digital displays, improving communication and operational transparency.

Priorities and Incremental Implementation:

Phase 1 (Initial Release):
Develop basic digital signage to display patient calls and general queue information.

Phase 2:
Integrate priority queue handling in the digital display system, ensuring urgent cases are attended to promptly.
Implement customization options for displays to accommodate specific departmental needs.

Phase 3:
Enhance the information display system with interactive features, such as estimated wait times and alerts for delays.

3 Domain model

<which information concepts will be managed in this domain? How are they related?>
<use a logical model (UML classes) to explain the concepts of the domain and their attributes>

4 Architecture notebook

4.1 Key requirements and constrains

<**Identify issues that will drive the choices for the architecture** such as: Will the system be driven by complex deployment concerns, adapting to legacy systems, or performance issues? Does it need to be robust for long-term maintenance?
Identify critical issues that must be addressed by the architecture, such as: Are there hardware dependencies that should be isolated from the rest of the system? Does the system need to function efficiently under unusual conditions? Are there integrations with external systems? Is the system to be offered in different user-interfacing platforms (web, mobile devices, big screens,…)?
E.g.: (the references cited in [XX] would be hypothetical links to previous specification documents/deliverables)

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

- The existing legacy Course Catalog System at Wylie College must be accessed to retrieve all course information for the current semester. The C-Registration System must support the data formats and DBMS of the legacy Course Catalog System [E2].
- The existing legacy Billing System at Wylie College must be interfaced with to support billing of students. This interface is defined in the Course Billing Interface Specification [E1].
- All student, professor, and Registrar functionality must be available from both local campus PCs and remote PCs with internet dial up connections.

- The C-Registration System must ensure complete protection of data from unauthorized access. All remote accesses are subject to user identification and password control.
- The C-Registration System will be implemented as a client-server system. The client portion resides on PCs and the server portion must operate on the Wylie College UNIX Server. [E2]
- All performance and loading requirements, as stipulated in the Vision Document [E2] and the Supplementary Specification [15], must be taken into consideration as the architecture is being developed.>

4.2 Architecture view

→ Discuss architecture planned for the software solution.
→ include a diagram (a package or block diagram)

→ explain how the identified modules will interact. Use sequence diagrams to clarify the interactions along time, when needed
→ discuss more advanced app design issues: integration with Internet-based external services, data synchronization strategy, distributed workflows, push notifications mechanism, distribution of updates to distributed devices, etc.>

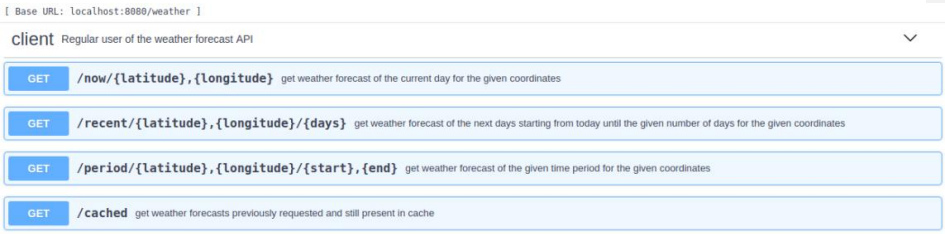
4.3 Deployment architecture

System Architecture
For the newPhi system, consider a three-tier architecture consisting of the presentation layer, the business logic layer, and the data access layer.

- **Presentation Layer:** This includes the *Patient* and *Desk* interfaces and digital signage (*Boards*). We will use html as framework.
- **Business Logic Layer:** This layer will handle all the business rules and processes. We will be using Spring Boot to manage the logic.
- **Data Access Layer:** Here, data interactions are handled. We will be using MongoDB.
- **Integration Layer:** Using RESTful APIs for communication between the front-end and the backend.

5 API for developers

[Explicar a organização da API. Os detalhes detalhes/documentação dos métodos devem ficar numa solução *hosted* de documentação de APIs, como o Swagger, Postman documentation, ou incluída no próprio desenvolvimento (e.g.: maven site)
<what services/resources can a developer obtain from your REST-API?>
<document the support endpoints>



6 References and resources

<document the key components (e.g.: libraries, web services) or key references (e.g.: blog post) used that were really helpful and certainly would help other students pursuing a similar work>