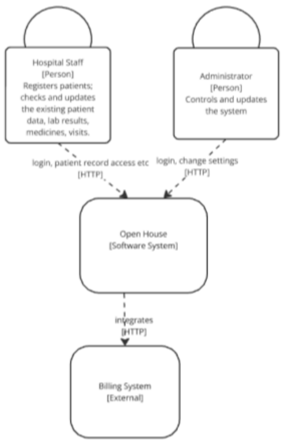
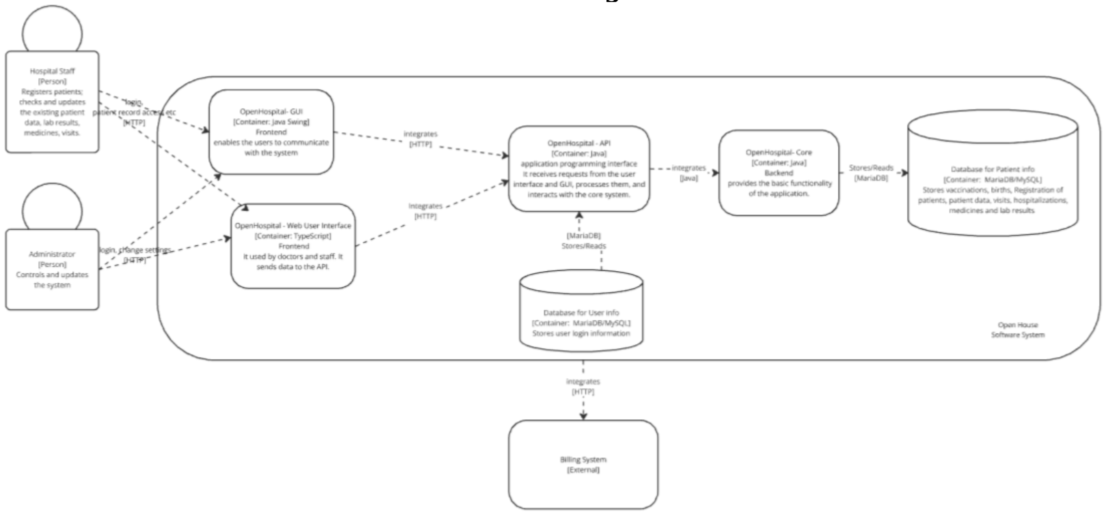
SOFT3215 - Software Architecture
Fall 2024 HW #1
22SOFT1027 Deniz Ziştoğlu - 21ARCH1017 Nurşen Eşan

1. Model the static aspects (from logical/structural viewpoint), and the dynamic aspects (from a behavioral viewpoint) separately, of the above mentioned software system with:
b. C4 (code level not required); two separate models, one for static aspects (logical/structural viewpoint:

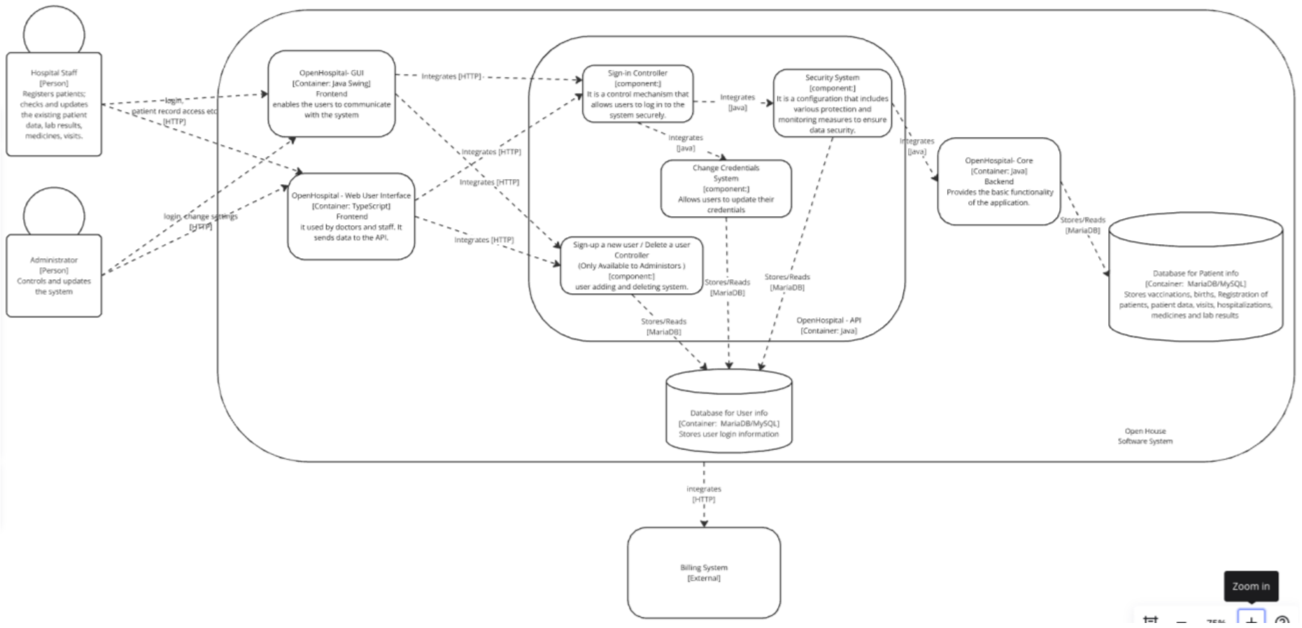
Context Diagram

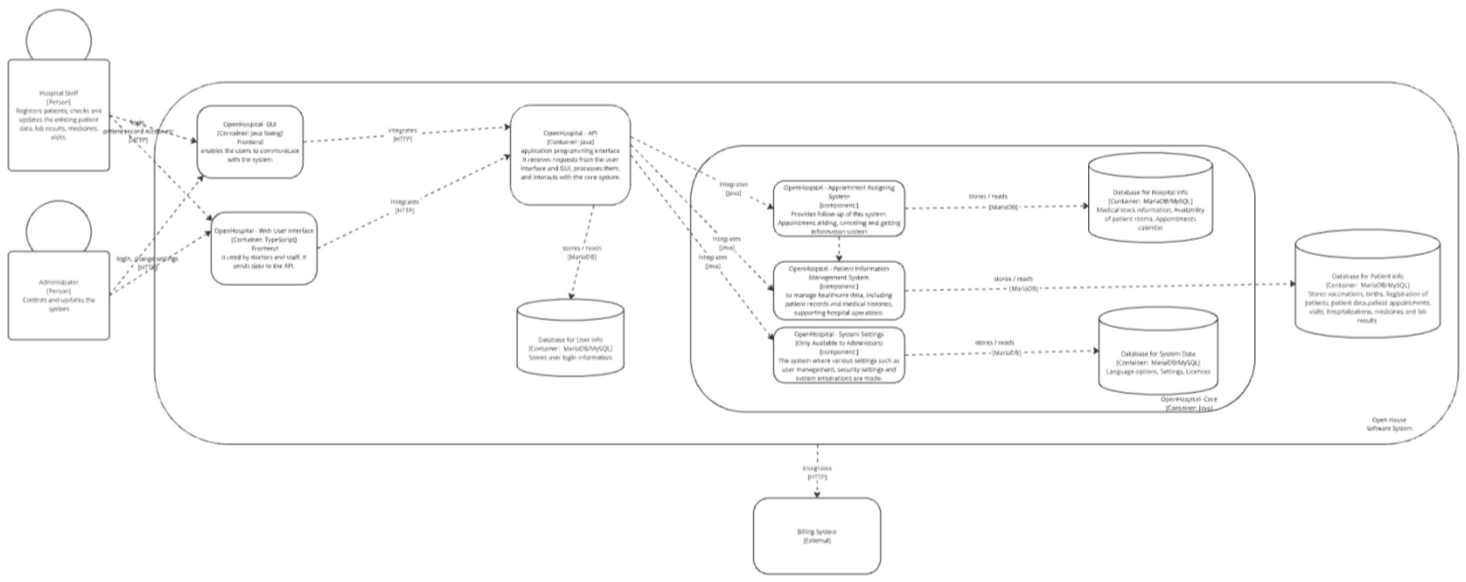

Hospital Staff
[Person]
Registers patients; checks and updates the existing patient data, lab results, medicines, visits.
Administrator
[Person]
Controls and updates the system
login, patient record access etc
[HTTP]
login, change settings
[HTTP]
Open House
[Software System]
integrates
[HTTP]
Billing System
[External]

Container Diagram:


Hospital Staff
[Person]
Registers patients; checks and updates the existing patient data, lab results, medicines, visits.
Administrator
[Person]
Controls and updates the system
login, patient record access etc
[HTTP]
login, change settings
[HTTP]
OpenHospital- GUI
[Container: Java Swing]
Frontend
enables the users to communicate with the system
OpenHospital - Web User Interface
[Container: TypeScript]
Frontend
It used by doctors and staff. It sends data to the API.
integrates
[HTTP]
Integrates
[HTTP]
OpenHospital - API
[Container: Java]
application programming interface
It receives requests from the user interface and GUI, processes them, and interacts with the core system.
integrates
[Java]
OpenHospital- Core
[Container: Java]
Backend
provides the basic functionality of the application.
Stores/Reads
[MariaDB]
Database for Patient info
[Container: MariaDB/MySQL]
Stores vaccinations, births, Registration of patients, patient data, visits, hospitalizations, medicines and lab results
[MariaDB]
Stores/Reads
Database for User info
[Container: MariaDB/MySQL]
Stores user login information
Open House
Software System
integrates
[HTTP]
Billing System
[External]

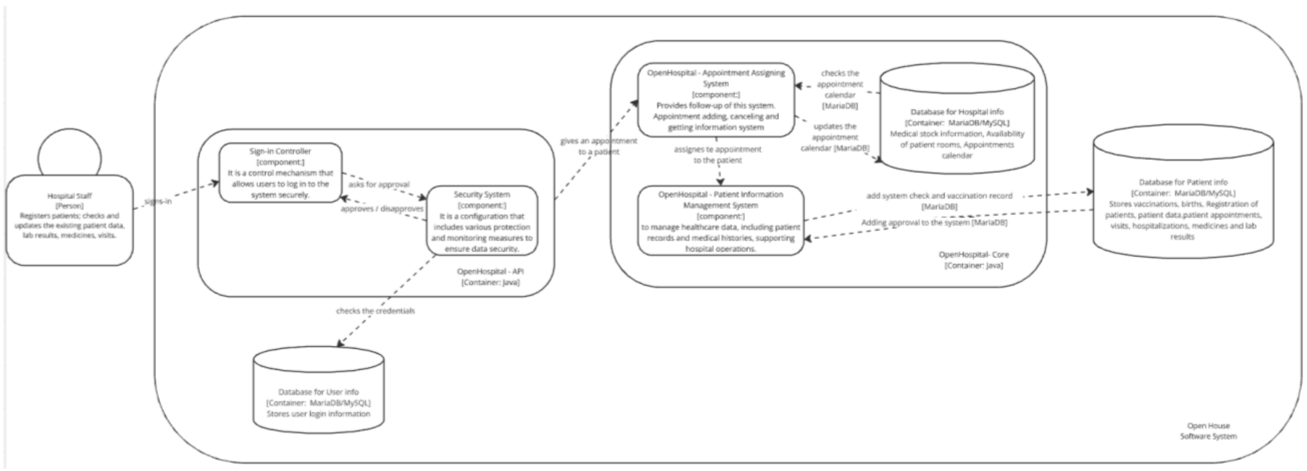
Component Diagrams:


Hospital Staff
[Person]
Registers patients; checks and updates the existing patient data, lab results, medicines, visits.
Administrator
[Person]
Controls and updates the system
login, patient record access etc
[HTTP]
login, change settings
[HTTP]
OpenHospital- GUI
[Container: Java Swing]
Frontend
enables the users to communicate with the system
OpenHospital - Web User Interface
[Container: TypeScript]
Frontend
it used by doctors and staff. It sends data to the API.
Integrates [HTTP]
Integrates [HTTP]
Integrates [HTTP]
Integrates [HTTP]
Sign-in Controller
[component:]
It is a control mechanism that allows users to log in to the system securely.
Integrates
[Java]
Security System
[component:]
It is a configuration that includes various protection and monitoring measures to ensure data security.
Integrates
[Java]
Change Credentials System
[component:]
Allows users to update their credentials
Sign-up a new user / Delete a user Controller
(Only Available to Administors)
[component:]
user adding and deleting system.
Stores/Reads
[MariaDB]
Stores/Reads
[MariaDB]
Stores/Reads
[MariaDB]
OpenHospital - API
[Container: Java]
Integrates
[Java]
OpenHospital- Core
[Container: Java]
Backend
Provides the basic functionality of the application.
Stores/Reads
[MariaDB]
Database for Patient info
[Container: MariaDB/MySQL]
Stores vaccinations, births, Registration of patients, patient data, visits, hospitalizations, medicines and lab results
Database for User info
[Container: MariaDB/MySQL]
Stores user login information
Open House
Software System
integrates
[HTTP]
Billing System
[External]
Zoom in


Hospital Staff
[Person]
Registers patients; checks and updates the existing patient data, lab results, medicines, visits.
Administrator
[Person]
Controls and updates the system
login, patient record access
[HTTP]
login, change settings
[HTTP]
OpenHospital- GUI
[Container: Java Swing]
Frontend
enables the users to communicate with the system
OpenHospital - Web User Interface
[Container: TypeScript]
Frontend
it used by doctors and staff, it sends data to the API.
integrates
[HTTP]
Integrates
[HTTP]
OpenHospital - API
[Container: Java]
application programming interface
It receives requests from the user interface and GUI, processes them, and interacts with the core system.
stores / reads
[MariaDB]
Database for User info
[Container: MariaDB/MySQL]
Stores user login information
Integrates
[Java]
OpenHospital - Appointment Assigning System
[component]
Provides follow-up of this system.
Appointment adding, canceling and getting information system
OpenHospital - Patient Information Management System
[component]
to manage healthcare data, including patient records and medical histories, supporting hospital operations.
OpenHospital - System Settings
(Only Available to Administrators)
[component]
The system where various settings such as user management, security settings and system integrations are made.
stores / reads
[MariaDB]
Database for Hospital info
[Container: MariaDB/MySQL]
Medical stock information, Availability of patient rooms, Appointments calendar
Database for Patient info
[Container: MariaDB/MySQL]
Stores vaccinations, births, Registration of patients, patient data,patient appointments, visits, hospitalizations, medicines and lab results
Database for System Data
[Container: MariaDB/MySQL]
Language options, Settings, Licences
OpenHospital- Core
[Container: Java]
Open House
Software System
Integrates
[HTTP]
Billing System
[External]

one for dynamic aspects (behavioral viewpoint):

This behavioral viewpoint shows a staff signing in and giving an appointment to a patient; after that checks the patient info and updates it accordingly.


Hospital Staff
[Person]
Registers patients; checks and updates the existing patient data, lab results, medicines, visits.
signs-in
Sign-in Controller
[component:]
It is a control mechanism that allows users to log in to the system securely.
asks for approval
approves / disapproves
Security System
[component:]
It is a configuration that includes various protection and monitoring measures to ensure data security.
OpenHospital - API
[Container: Java]
checks the credentials
Database for User info
[Container: MariaDB/MySQL]
Stores user login information
gives an appointment to a patient
OpenHospital - Appointment Assigning System
[component:]
Provides follow-up of this system.
Appointment adding, canceling and getting information system
checks the appointment calendar [MariaDB]
updates the appointment calendar [MariaDB]
Database for Hospital info
[Container: MariaDB/MySQL]
Medical stock information, Availability of patient rooms, Appointments calendar
assignes te appointment to the patient
OpenHospital - Patient Information Management System
[component:]
to manage healthcare data, including patient records and medical histories, supporting hospital operations.
add system check and vaccination record [MariaDB]
Adding approval to the system [MariaDB]
Database for Patient info
[Container: MariaDB/MySQL]
Stores vaccinations, births, Registration of patients, patient data,patient appointments, visits, hospitalizations, medicines and lab results
OpenHospital- Core
[Container: Java]
Open House
Software System

2. Compare and contrast your experiences from the first part above, and note what kinds of information were easy, hard, or impossible to capture and model in either of the notations. Explain how you established the consistency among the notations?

When modeling the Open Hospital system, we found that some aspects were easy to capture, while others presented significant challenges. High-level interactions between users, could be easily identified and represented in the Context Diagram, clearly showing their relationship to the system. Additionally, relationships between large containers, such as the UI, API, and Databases, were easily visible, making it easier to create the Container Diagram. However, it was more difficult to describe interactions between internal components in detail, as documentation was limited. Understanding workflows, especially complex processes involved in appointments, required a deep dive into the codebase. Some implementation details, such as how the system managed concurrent requests were difficult to capture accurately, reflecting a lack of clarity in existing documentation. To maintain consistency across diagrams, we cross-referenced them and components to ensure that a unified terminology was used. We tried to accurately represent dynamic behaviors. We experienced some difficulties while modeling complex systems, where comprehensive architectural documentation is lacking.

3. What was available in the project web site with regard to architectural documentation of the system? What was missing? How did this affect your work for this assignment?

The architectural documentation for the Open Hospital project provided a basic overview of the system's goals and core functionality. The website included brief descriptions of major modules and provided insights into their intended operations. However, these descriptions were not sufficient; the documentation fell short in several critical areas, particularly the lack of detailed architectural diagrams depicting the overall system structure and data flow. This lack of representation made it difficult to clearly grasp the relationships between components. Additionally, there was insufficient information about component interactions and state management processes, which complicated the modeling of dynamic behaviors. Although the GitHub repository provided access to the source code and some contextual information, extensive analysis was required to extract the architectural design. As a result, these problems led us to rely on reasonable assumptions.

4. If you were a project manager for a software project, how much of a project's budget would you devote to software architecture documentation? Why? How would you measure the cost and the benefit?

Because the Open Hospital is an open sourced software intended for long-term development and can be contributed by many developers; software architecture documentation being clear, understandable and overall proper is quite important. Clear architectural documentation would ensure that all team members and contributors understand the system's structure, this would reduce the risk of conflicting implementations and overall confusion.

Considering the budgeting of similar projects, we would devote approximately 10-15% of the project's budget to architecture documentation. This approximation might vary depending on factors like the project's complexity, amount of team members, and expected lifecycle.