Indiana University-Purdue University Indianapolis
INFO-C451 System Implementation
Instructor Shawn Dai

HealthCare Management System

System Design, v2

Kayla Heydon

TABLE OF CONTENTS

1. SYSTEM ARCHITECTURE AND SYSTEM DESIGN

System architecture is a scheme that describes a system's behavior, structure and views of the system. The architecture can consist of system and subsystem.

1.1. Architectural Styles

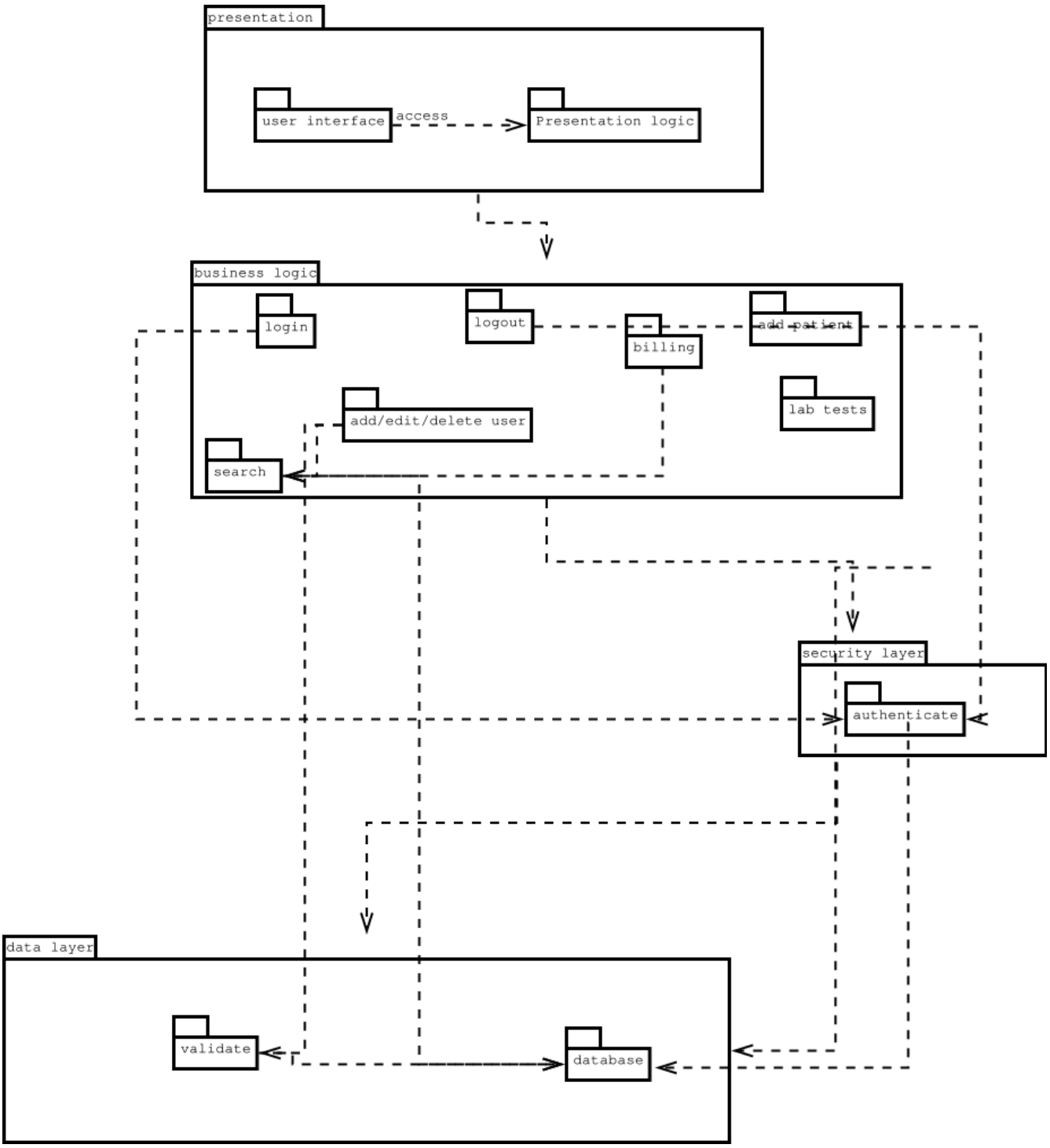
The system uses a client-server architecture, thus allowing the system to be distributed within the facility, take, for example, the administrator can have his/her client that interfaces with the server. This architectural style allows the server to be in a centralized place, and the client can be in different departments within the hospital facility. This works to our advantage because it allows the system to grow both horizontally and vertically. In the future, if the hospital needs to add a new department, all that is need is to connect the client to the server. To future make the system flexible both the layered and component-based approach was used this allows the system functions to be sub-divided adhering to the design principles of the project which state that each class within the system can only handle one responsibility take for example the search class which implements the search functionality in the system the class can the accessed by other classes needing the search functionality.

The server side is divided into three tiers

1. The business logic unit oversees all the business logic, for example, calculating the bill
2. The database is in cost of storing all the hospital-related data
3. The authentication and authorization unit handle all the sessions within the system, thereby providing security.

1.2. Identifying Subsystems

The system will be made up of a front-end GUI who role is to interface with the end-user to enable for interaction with the system and viewing of the data from the system.
The frond end server on the other hand retrieves data from the database and avails it to the user through the front-end GUI. The front-end server also stores new data in the database. The database is tasked with holding persistent information on patients and staff. The backend on the other hand provides feedback to doctors and administrators and is also used for updating user information. It can also be used to access data histories. The figure below shows the systems subsystems.



1.3. Mapping Subsystems to Hardware

Our system architectural style is majorly based on the server-client model, the presentation subsystem will sit the client-side, and the rest of the subsystems will sit on the server-side. On the client-side, the standard user needs to have a personal computer running any browser. On the server-side, we need to have two bare metal servers. The first one runs our data subsystem, while the second runs the

authentication and business logic subsystems. This setup allows the system to run faster because of the distribution of tasks. The configuration further boots security because of the layering of security.

1.4. Persistent Data Storage

The system uses a relational database to store data. The data split into tables following the entity relation diagram.

- staff table
- patient table
- billing table
- room table

Each of these tables has its attributes that describe the entities in these tables.

1.5. Network Protocol

The server side uses sockets to connect between the database and the business logic subsystems, each running on a different server. The network protocol used is TCP protocol, which allows end to end connection between the two. The relationship between the client and the server uses the secure https protocol. In the https protocol, the data is encrypted.

1.6. Global Control Flow

The system is event driven. The server is always looking for connections when a client connects the server. It creates a thread to handle the needs of that client. It continues to look for incoming connections, because of threading the different clients can execute various tasks within the system, which removes the time dependency. Also, in the data subsystem, the database spins off threads to handle the different requests by the client through the business logic subsystem. To maintain the integrity of the system, the threads make use of semaphores, mutexes, and locks. These controls prevent two threads from writing to the same object at the same time, hence corrupting the data.

1.7. Hardware Requirements

The client-side does not need any specific hardware to run a standard personal computer should suffice, but on the server-side, we need to use machines with the following requirements;

- CPU with at least six cores
- 16 GB of RAM
- One Gigabyte capable network card
- 1 Terabyte solid-state drive
- 4 Terabytes hard disk

The above requirements will make the system have a faster response time because the CPU can handle all the incoming client requests. Also, the RAM makes the machine handle a lot of processes at a time. This also allows users to fully maximize the CPU, for the one-gigabyte solid-state drive this make our reads and write to the database even faster. The network card allows the system to use internet connections of over one gigabyte, thereby increasing the overall response time of the system.

2. REFERENCES

Cohn, Mike (2004). *User Stories Applied for Agile Software Development*. Addison Wesley.

Cohn, Mike (2005). *Agile Estimating and Planning*. Addison-Wesley.

Ribu, Kirsten (2001). *Estimating Object-Oriented Software Projects with Use Cases*. Master of Science Thesis, University of Oslo, Department of Informatics.

Schneider, Geri and Jason P. Winters (1998). Applying Use Cases: A Practical Guide. Addison Wesley.

Schwaber, Ken and Mike Beedle (2001). *Agile Software Development with Scrum*. Prentice Hall.