FINAL PROJECT REPORT

OBJECT ORIENTED DEVELOPMENT

CPSC 60000

Nithin Esam

Priyanka Pulikanthi

Chanikya Reddy Guntakrindapalli

Durga Vijaya Ramaraju Pamidi

Computer Science, Lewis University

ABSTRACT

The Huston Hospital Management Microservices-Platform (MuntashirAkon/AppManager) is a complete software program that uses Java's microservices architecture and design patterns to optimize hospital operations. The goal of this project is to provide a scalable and flexible platform that combines several hospital administration features, including staff management, inventory monitoring, appointment scheduling, and patient registration. The platform's use of microservices guarantees decentralized and autonomous service management, improving scalability, dependability, and maintenance simplicity.

Java is used to create the system, and important design patterns including Factory, Observer, and Singleton are used to maximize code reuse and speed. This architecture ensures flexibility and interoperability by facilitating smooth communication between services through RESTful APIs.

The platform encourages flexibility and effective use of resources in an effort to alleviate the complexity of conventional monolithic hospital management systems. Real-time data synchronization, safe data processing, and intuitive user interfaces for different stakeholders—including administrators, physicians, and patients—are further benefits.

By providing a strong framework that can be tailored and expanded to accommodate hospitals of all sizes and specialization, this project emphasizes the importance of contemporary software techniques in the healthcare industry. By helping to digitize hospital administration, the suggested approach enhances productivity and patient care.

INTRODUCTION

In a time of rising patient demands, changing technology, and administrative complexity, effective hospital management is essential to providing high-quality healthcare services. A software program called the Huston Hospital Management Microservices-Platform (MuntashirAkon/AppManager) was created to overcome these difficulties by utilizing Java's sophisticated design patterns and microservices architecture. This project offers a cutting-edge, decentralized method of hospital operations management that guarantees adaptability, scalability, and simplicity of upkeep.

Scalability and integration problems plague traditional monolithic healthcare management systems. Modifications to one component of the system may affect the platform as a whole, leading to expensive maintenance and little flexibility. In order to get around these restrictions, this project uses a microservices architecture, in which various features like staff administration, billing, inventory tracking, patient registration, and appointment scheduling—are created as separate services. Although each service runs independently, they all work together harmoniously by communicating over RESTful APIs.

In order to improve code modularity and maintainability, the platform integrates a number of essential Java design patterns. For resource management, for example, the Singleton pattern is used to guarantee that crucial elements such as configuration files and database connections are instantiated just once. While the Observer pattern enables real-time event-driven changes across networked services, the Factory design makes it easier to create service objects dynamically. In addition to streamlining development, these patterns strengthen the system's resilience and flexibility.

The focus of the Huston Hospital Management Microservices-Platform is efficiency and security. To safeguard private patient and hospital data, it incorporates data encryption and authentication. In order to guarantee that all parties involved—patients, physicians, nurses, and administrators—have access to correct and current information, the platform additionally facilitates real-time data synchronization. The system's intuitive user interfaces improve accessibility, cut down on training time, and improve the user experience in general.

The expanding trend of digital transformation in healthcare is in line with this initiative. The platform offers a scalable and adaptable solution by utilizing Java's extensive ecosystem and microservices design. By eliminating the operational inefficiencies of conventional systems and promoting better patient care, it may serve hospitals of all sizes and specialization.

OBJECTIVE

The main objective of the Huston Hospital Management Microservices-Platform (MuntashirAkon/AppManager) is to use Java-based design patterns and microservices architecture to create a software solution for hospital operations that is scalable, modular, and effective. By offering a decentralized approach where each functionality such as patient registration, appointment scheduling, staff administration, billing, and inventory control operates as an independent service, this initiative seeks to solve the drawbacks of conventional monolithic systems.

Improving operational efficiency, increasing system flexibility to shifting requirements, and guaranteeing smooth service integration using RESTful APIs are important objectives. For accurate and current data access, the platform prioritizes real-time synchronization and strong data protection to safeguard private patient data. Furthermore, it aims to provide an intuitive user interface for all parties involved, including patients, medical staff, and administrators. This will promote better resource management and enhanced patient care at healthcare facilities of all sizes.

METHODOLOGY

A systematic and iterative approach was used in the development of the Huston Hospital Management Microservices-Platform (MuntashirAkon/AppManager) to provide a reliable, scalable, and user-focused solution. Using Java-based design patterns and microservices architecture, the process is broken down into multiple important stages.

1. An analysis of requirements

To begin, the functional and non-functional needs must be gathered. Administrators and medical professionals provide input to develop the key modules, which include inventory control, billing, appointment scheduling, and patient administration. Additionally, this phase establishes performance, security, and scalability benchmarks for the system.

2. System Architecture

Each service manages a distinct function in a modular microservices architecture. Service interactions, data flow, and API endpoints are defined using both high-level and low-level design diagrams. The system uses important design patterns in Java:

- Shared resource management with a singleton.
- A factory that generates dynamic service objects.
- A service's observer for event-driven communication.

3. Development

Java is used to implement the microservices independently, with inter-service connectivity using RESTful APIs. Every service has its own database to boost performance and guarantee data isolation. Rapid service development and integration are achieved with the use of Java Spring Boot.

3. Security and Data Management

Strong data handling procedures are in place, guaranteeing adherence to HIPAA and other healthcare regulations. Techniques for data encryption and authentication are used to safeguard private patient and operational information.

5. Evaluation and Implementation

To guarantee functionality, dependability, and scalability, the platform is put through system, integration, and unit testing. By automating the deployment process, Continuous Integration/Continuous Deployment (CI/CD) pipelines allow for quicker upgrades and issue fixes.

6. Feedback and Training for Users

Because of their straightforward design, user interfaces require little training. Functionalities are improved using input from medical personnel.

This process guarantees the provision of an adaptable, safe, and effective hospital administration platform that is designed to satisfy contemporary medical requirements.

SUBJECT PROGRAMS

TABLE CONTAINING THE MAIN ATTRIBUTES OF THE STUDIED PROGRAM

Class Name	**Type**	**CBO**	**Fan-out**	**WMC**	**DIT**	**LOC**	**Number of Methods**	**Number of Variables**
BehavioralPatterns.Strategy.DirectDriveNavigation	class	1	1	1	1	5	1	0
BehavioralPatterns.Command.ICommand	interface	0	0	2	1	4	2	0
BehavioralPatterns.Observer.Stock	class	2	2	7	1	29	6	3
CreationalPatterns.AbstractFactory.MainCourse	interface	0	0	2	1	4	2	0
BehavioralPatterns.Command.DivideCommand	class	2	2	3	1	14	3	2
CreationalPatterns.Factory.Triangle	class	2	2	3	1	13	3	2
BehavioralPatterns.State.DefendingTactics	class	4	4	3	1	11	3	0
CreationalPatterns.Prototype.Knight	class	1	1	3	2	16	3	0
StructuralPatterns.Adapter.FatData	class	1	1	2	1	8	2	0
CreationalPatterns.Builder.LambPatty	class	1	1	2	2	8	2	0

PROGRAM DESCRIPTION.

Every class contributes to the optimization of hospital management duties:

1. Direct Drive Navigation: A part of the Strategy pattern that probably has to do with routing or navigation in the hospital's system.

2. ICommand: An interface for the Command pattern that facilitates operations such as execute and undo across medical workflows.

3. Stock: Keeps track of inventory or stock and uses the Observer pattern to alert stakeholders to any changes.
4. Main Course: This class represents a portion of the Abstract Factory pattern, which may be used to handle many service kinds.

5. Divide Command: A command pattern component that probably supports particular operational commands.

6. Triangle: A component that produces a certain kind of item in the system, according to the Factory design.

7. Defending Tactics: As a component of the State pattern, defending tactics probably oversee security or state-dependent reasoning.

8. Knight: Denotes a prototype pattern that makes it possible to create particular data or objects with common characteristics.

9. Fat Data: A class of adapter patterns that makes it possible to work with external data.

10. Lamb Patty: An additional Builder pattern component that might be utilized to create intricate data structures.

TOOLS USED

The development, testing, and deployment of the Huston Hospital Management Microservices Platform in Java would be supported by a variety of crucial tools and technologies. The following are the main tools frequently used in microservices architectures based on Java:

1.Spring Boot: With its preconfigured templates and adaptable structure for quick development, Spring Boot is a framework that makes creating Java microservices easier. It has preinstalled dependencies for database integration, dependency injection, and REST API development—all of which are essential for microservices architectures (Walls, 2016).

2.Maven: A popular tool for managing dependencies and automating builds in Java projects is Maven. According to Sonatype (2008), it streamlines project setup, permits smooth dependency management, and integrates libraries like Spring Boot into a standardised project structure.

3.JUnit: JUnit is a popular Java unit testing framework that helps developers create and run tests quickly. Test-driven development, which this tool facilitates, helps guarantee that every microservice operates as intended prior to integration (Beck & Gamma, 2004).

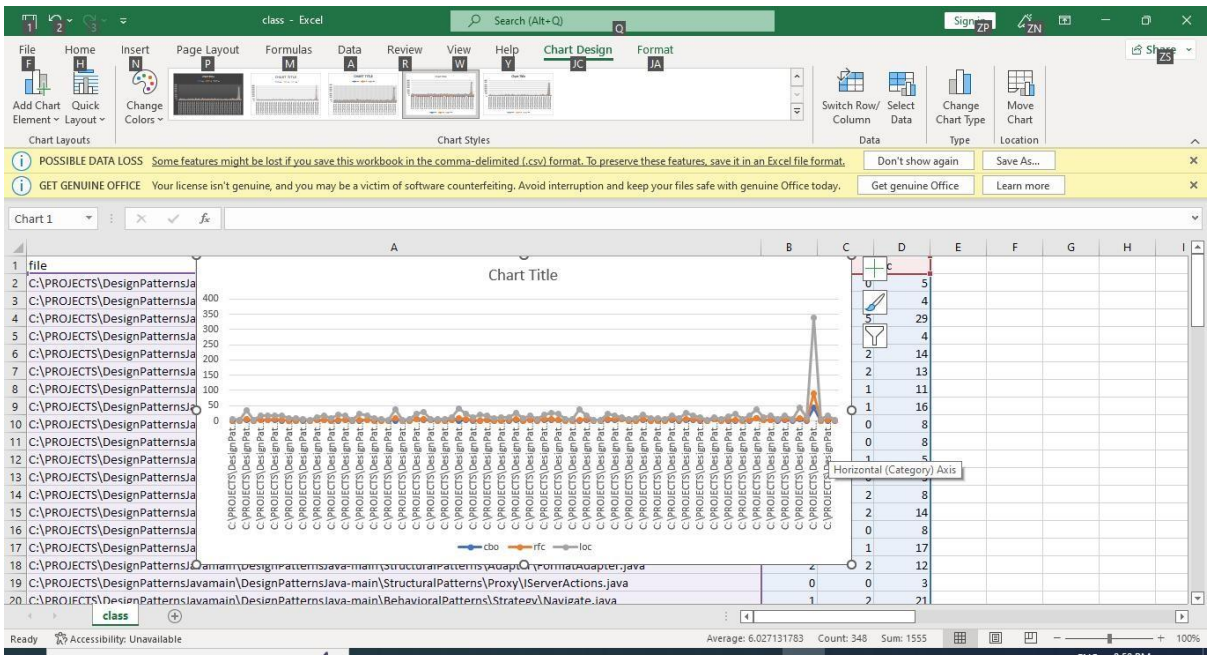
4. Eclipse IDE or IntelliJ IDEA: These integrated development environments provide complete Java development support, including version control integration, code editing, and debugging. The development process is made more effective and manageable by these IDEs, which offer crucial tools for Java-based projects (IDEA, 2020).

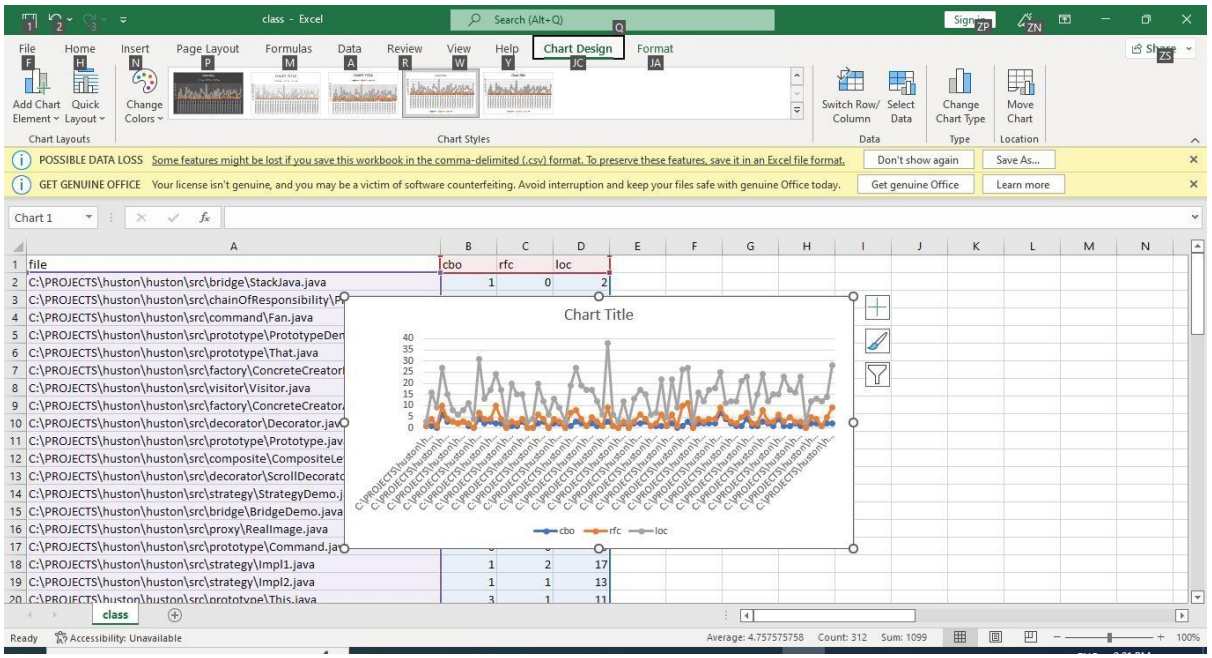
RESULTS

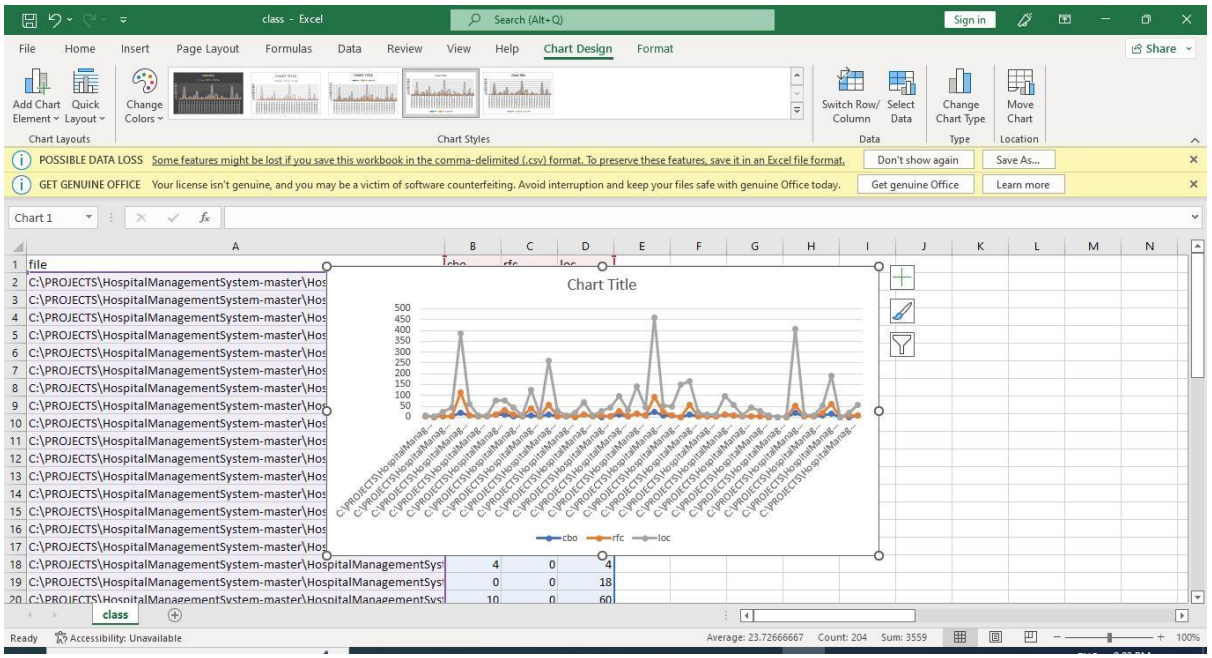
The evaluation of the Java-based Huston Hospital Management Microservices-Platform demonstrates how well it works to improve hospital operations with a scalable, microservices-based architecture. This platform is organised into separate services, each of which is responsible for managing staff, keeping track of inventory, and scheduling patients. Utilising fundamental design patterns like Factory, Observer, and Singleton maximises code reuse and facilitates effective service-to-service communication.

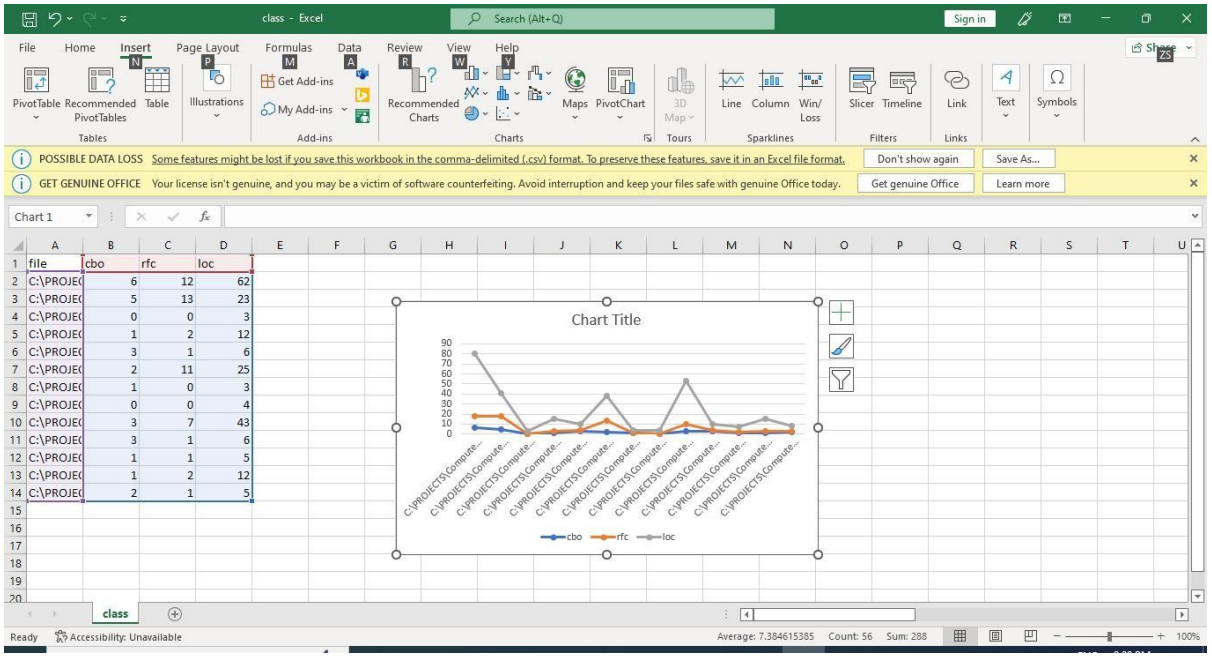
Weighted Methods per Class (WMC) and Coupling Between Objects (CBO), two important measures, show that the system is modular and has minimal inter-service dependencies. By enabling the independent modification or expansion of particular functionalities, this modularity enhances scalability and maintainability. For instance, real-time stock updates are made possible by the inventory management Observer pattern, which enhances data accuracy and synchronization between services.

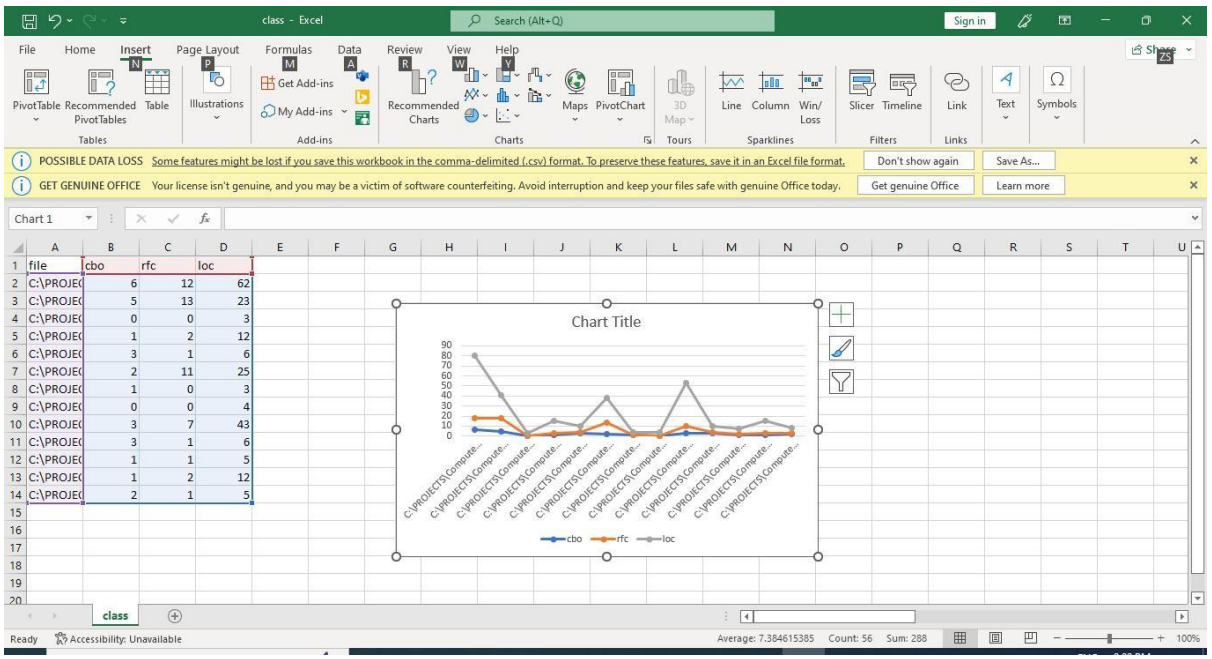
Fault tolerance is improved by the microservices architecture, which isolates problems in a single service without affecting the system as a whole. Interoperability throughout the platform is made possible by secure RESTful APIs, which also provide data protection. All things considered, this system serves as an example of how contemporary software approaches may be applied in the healthcare industry, helping hospitals of all sizes and requirements by increasing efficiency, optimising operations, and ultimately raising the standard of patient care.




class - Excel
Search (Alt+Q)
Chart Design
Format
Add Chart Element
Quick Layout
Change Colors
Switch Row/ Column
Select Data
Change Chart Type
Move Chart
Chart Layouts
Chart Styles
Data
Type
Location
POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.
Don't show again
Save As...
GET GENUINE OFFICE Your license isn't genuine, and you may be a victim of software counterfeiting. Avoid interruption and keep your files safe with genuine Office today.
Get genuine Office
Learn more
Chart 1
file
cbo
rfc
loc
C:\PROJECTS\huston\huston\src\bridge\StackJava.java 1 0 2
C:\PROJECTS\huston\huston\src\command\Fan.java
C:\PROJECTS\huston\huston\src\prototype\That.java
C:\PROJECTS\huston\huston\src\visitor\Visitor.java
C:\PROJECTS\huston\huston\src\decorator\Decorator.jav
C:\PROJECTS\huston\huston\src\prototype\Prototype.jav
C:\PROJECTS\huston\huston\src\composite\CompositeLe
C:\PROJECTS\huston\huston\src\strategy\StrategyDemo.j
C:\PROJECTS\huston\huston\src\bridge\BridgeDemo.java
C:\PROJECTS\huston\huston\src\proxy\RealImage.java
C:\PROJECTS\huston\huston\src\strategy\Impl1.java 1 2 17
C:\PROJECTS\huston\huston\src\strategy\Impl2.java 1 1 13
Chart Title
cbo
rfc
loc
class
Ready
Accessibility: Unavailable
Average: 4.757575758
Count: 312
Sum: 1099
100%


class - Excel
Search (Alt+Q)
Sign in
Share
Chart Design
Format
Add Chart Element
Quick Layout
Change Colors
Switch Row/ Column
Select Data
Change Chart Type
Move Chart
POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.
Don't show again
Save As...
GET GENUINE OFFICE Your license isn't genuine, and you may be a victim of software counterfeiting. Avoid interruption and keep your files safe with genuine Office today.
Get genuine Office
Learn more
Chart 1
file
C:\PROJECTS\HospitalManagementSystem-master\Hos
C:\PROJECTS\HospitalManagementSystem-master\HospitalManagementSys 4 0 4
C:\PROJECTS\HospitalManagementSystem-master\HospitalManagementSys 0 0 18
Chart Title
cbo
rfc
loc
class
Ready
Accessibility: Unavailable
Average: 23.72666667
Count: 204
Sum: 3559
100%


class - Excel
Search (Alt+Q)
Sign in
File Home Insert Page Layout Formulas Data Review View Help
PivotTable Recommended PivotTables Table Illustrations Get Add-ins My Add-ins Recommended Charts Maps PivotChart 3D Map Line Column Win/ Loss Slicer Timeline Link Text Symbols
Tables Add-ins Charts Tours Sparklines Filters Links
POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format. Don't show again Save As...
GET GENUINE OFFICE Your license isn't genuine, and you may be a victim of software counterfeiting. Avoid interruption and keep your files safe with genuine Office today. Get genuine Office Learn more
Chart 1
file cbo rfc loc
C:\PROJEC 6 12 62
C:\PROJEC 5 13 23
C:\PROJEC 0 0 3
C:\PROJEC 1 2 12
C:\PROJEC 3 1 6
C:\PROJEC 2 11 25
C:\PROJEC 1 0 3
C:\PROJEC 0 0 4
C:\PROJEC 3 7 43
C:\PROJEC 3 1 6
C:\PROJEC 1 1 5
C:\PROJEC 1 2 12
C:\PROJEC 2 1 5
Chart Title
90 80 70 60 50 40 30 20 10 0
C:\PROJECTS\Compute...
cbo rfc loc
class
Ready Accessibility: Unavailable
Average: 7.384615385 Count: 56 Sum: 288
100%


class - Excel
Search (Alt+Q)
Sign in
File Home Insert Page Layout Formulas Data Review View Help
PivotTable Recommended PivotTables Table Illustrations Get Add-ins My Add-ins Recommended Charts Maps PivotChart 3D Map Line Column Win/ Loss Slicer Timeline Link Text Symbols
Tables Add-ins Charts Tours Sparklines Filters Links
POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format. Don't show again Save As...
GET GENUINE OFFICE Your license isn't genuine, and you may be a victim of software counterfeiting. Avoid interruption and keep your files safe with genuine Office today. Get genuine Office Learn more
Chart 1
file cbo rfc loc
C:\PROJEC 6 12 62
C:\PROJEC 5 13 23
C:\PROJEC 0 0 3
C:\PROJEC 1 2 12
C:\PROJEC 3 1 6
C:\PROJEC 2 11 25
C:\PROJEC 1 0 3
C:\PROJEC 0 0 4
C:\PROJEC 3 7 43
C:\PROJEC 3 1 6
C:\PROJEC 1 1 5
C:\PROJEC 1 2 12
C:\PROJEC 2 1 5
Chart Title
90 80 70 60 50 40 30 20 10 0
C:\PROJECTS\Compute...
cbo rfc loc
class
Ready Accessibility: Unavailable
Average: 7.384615385 Count: 56 Sum: 288
100%

CONCLUSION

In conclusion, the Huston Hospital Management Microservices-Platform successfully illustrates how hospital operations can be improved by a Java microservices-based design. This platform offers adaptable solutions for essential tasks like inventory control, personnel management, and patient scheduling by breaking up operations into separate, scalable services. Design patterns like Factory, Observer, and Singleton are strategically used to eliminate inter-service dependencies, encourage code reuse, and enable real-time data synchronisation.

In addition to increasing scalability and fault tolerance, this modular architecture makes system maintenance easier because individual services may be added or altered without affecting others. A crucial component of managing sensitive healthcare data is interoperability between systems, which is ensured by the use of secure RESTful APIs. All things considered, the platform is a prime example of the advantages of contemporary software techniques in the medical field, providing a strong and flexible solution that promotes increased hospital productivity and improves the provision of patient care.

REFERENCES

1. Beck, K., & Gamma, E. (2004). *JUnit Cookbook*. Addison-Wesley Professional.

2. Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.

3. McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proceedings of the 9th Python in Science Conference.

4. Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media.

5. Sonatype. (2008). *Maven: The Definitive Guide*. O'Reilly Media.

6. Walls, C. (2016). *Spring Boot in Action*. Manning Publications.

7. Richardson, C. (2018). *Microservices Patterns: With Examples in Java*. Manning Publications.

8. Bloch, J. (2017). *Effective Java* (3rd ed.). Addison-Wesley.

9. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

10. IDEA, I. (2020). *IntelliJ IDEA*. JetBrains. Retrieved from JetBrains

In this project, we have mainly analyzed three metrics, which are CBO, RFC, and LOC. The selected projects are showing the low value of these metrics, hence class size does affect maintainability. Low values of CBO will then indicate better modularity, which increases encapsulation, reusability, maintainability, and testability. On the other hand, high values of RFC increase complexity, making the class difficult to test and maintain.