JAVA UPGRADER

A PROJECT REPORT

Submitted by

ROBIN MONACHAN

REG NO: TKM20MCA-2032

to

The APJ Abdul Kalam Technological University

In partial fulfillment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Thangal Kunju Musaliar College of Engineering
Kerala

DEPARTMENT OF COMPUTER APPLICATIONS

JULY 2022

DECLARATION

I **ROBIN MONACHAN** hereby declare that the project report "**JAVA UPGRADER"**, submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Application of the **APJ Abdul Kalam Technological University, Kerala** is a bonafide work done by me under supervision **of Prof. NADERA BEEVI S.**

This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources.

I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Kollam **ROBIN MONACHAN**

18-07-2022

DEPARTMENT OF COMPUTER APPLICATION
THANGAL KUNJU MUSALIAR COLLEGE OF ENGINEERING

C E R T I F I C A T E

This is to certify that the report entitled **JAVA UPGRADER** submitted by **ROBIN MONACHAN (TKM20MCA-2032)**, to the APJ Abdul Kalam Technological University in partial fulfillment of the degree in Master of Computer Applications is a bonafide record of the project work carried out by her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor Head of the Department External Examiner

SUN/HRD/2022/06-10/026
10 June 2022

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Robin Monachan (EMP ID: T30029**) has been employed with us from 19 January 2022 to 10 June 2022.

His designation is **"Project Intern"** at the time of leaving the organization.

We wish him all success in his future endeavors!

for **SunTec Digital Solutions Private Limited,**

Prakash P Nair
Head – Human Resources

ACKNOWLEDGEMENT

First and foremost I thank GOD almighty and my parents for the success of this project. I owe sincere gratitude and heart full thanks to everyone who shared their precious time and knowledge for the successful completion of my project.

I am extremely grateful to ***Prof. Dr. Fousia M Shamsudeen***, Head of the Department, Department ofComputer Applications, for providing me with best facilities.
I would like to thank our coordinator and project guide ***Prof. Vaheetha Salam***
Department of Computer Applications, who motivated me throughout the project
.

I would like to thank my external coordinator ***Mr. Vinoth Kumar Muppudathy***, Suntec Business Solution, who guided me throughout my work.

I profusely thank all other faculty members in the department and all other members of TKM Collegeof Engineering, for their guidance and inspirations throughout my course of study.

I owe my thanks to my friends and all others who have directly or indirectly
helped me in the successful completion of this project.

ROBIN MONACHAN

ABSTRACT

A few years ago, some projects were started with an older version of Java (LTS), which was the most stable version of the language and virtual machine at the time. But as time went on, new Java(LTS) became the official and latest LTS version, so we need to start moving our projects to that version.

One difference between Java version 8 and version 11 is that some features your dependencies need may have been taken out of the SDK's core packages. If you do that, you will get errors like "Package X cannot be found." To handle these edge cases, we need to figure out what extra dependencies we may need to add, add them to the pom.xml file, and change the import lines that go with them.

This project is mostly used to change all the files in a project from their old values to their new values. The old content is read by this project, and the new content is put in its place. This is like making a change from JDK1.8 to JDK11. This will also change which libraries are used when something is imported into a programme.

Contents

List of Figures

CHAPTER 1
INTRODUCTION

According to the 2021 Java Developer Productivity Report, 69% of developers are still using Java 8 - and the majority have not updated since Java 8's release in 2014. People wonder how important it is to update since Java 17 is the latest LTS version. A piece of technology changes when it is updated. New features, security patches, the removal of old features, and more support are all part of Java LTS releases. This means that anyone who hasn't updated since Java 8 or after should really think about this.

Some projects were started a few years ago with an older version of Java (LTS), which at the time was the most stable version of the language and virtual machine. But as time went on, new Java(LTS) became the official and most recent LTS version, so we need to start moving our projects to that version.

Even if you don't care about the new features, using an older version of Java could put your security at risk in the future. Non-LTS releases are only supported for six months after they are made available to the public. After that, they don't get any more public news. Each new SE release has major security fixes, but as you can see, this means that each new release needs to be updated every six months.

Your team might be better off in the long run if you spend money once. If your team upgrades to an LTS version with a guaranteed shelf life of at least three years, they won't have to worry about security patches and updates every six months.

One difference between Java 8 and Java 11 is that some features that your dependencies need may have been taken out of the SDK's core packages. If you do that, you'll get errors like "Package X cannot be found." To handle these edge cases, we need to figure out what extra dependencies we might need to add, add them to the pom.xml file, and change the import lines that go with them.

Most of the time, this project is used to change the old values of all the files in a project to their new values. This project reads the old content and replaces it with the new content. It's the same

as switching from JDK1.8 to JDK11. When something is imported into a programme, this will also change which libraries are used.

From a technical point of view, it takes time to learn how to upgrade. For example, if you use the Nashhorn JavaScript engine, which was available in Java versions before 15, you should know that it won't be available in Java 17, so you'll have to find a replacement.

When your team upgrades to an LTS version, they get new features and better support, but it takes time and makes them slower at first. Updating your Java application infrastructure does take time and money, and the more Java apps you have running, the longer it takes. Also, you'll need to make sure that any third-party apps you use will work with the new version. If your team is still using Java 8 or earlier, you are missing out on many new features and functions, like JDK enhancement processes (JEP). Change and new ideas, which are still going strong, have helped Java move forward. Java will probably stay a top business language for a long time. Since help for older versions of Java is getting harder to find, those who upgrade sooner will be ahead of those who still don't want to move.

1.1 COMPANY PROFILE

SunTec is the best company in the world when it comes to prices and billing. It helps businesses with its cloud-based products. SunTec is trusted by more than 130 clients in 45+ countries to provide hyper-personalized products, offers, pricing, loyalty programmes, and billing for more than 400 million end customers. SunTec products are built on our API-first, micro-services-based, cloud-native, cloud-agnostic platform, Xelerate, and are delivered on-premise, on private cloud, and as SaaS. SunTec has offices in many countries, including the US, UK, Germany, UAE, Singapore, Canada, Australia, and India. If you want to know more.

Our Xelerate platform, which is cloud-native and cloud-agnostic, API-first, and based on microservices, is what the SunTec products are built on. Our products can be delivered on-site, in a private cloud, or as SaaS. They have high performance, advanced security, scalability, resilience, and agility, even when they have to deal with a lot of customer and transaction data. Our products can be linked to systems for doing business, getting new customers, and keeping

track of partners. They can also be used to show products, deals, and prices to customers and business partners in real time.

Enterprise Product Management :

SunTec Enterprise Product Management offers companies with a technique that is both consistent and scalable, allowing for the production and deployment of highly individualised, cutting-edge products and services that are customised precisely to a variety of customer groups. It helps in the establishment of a single central repository for in-house as well as partner goods and services, the standardisation of end-to-end product and service lifecycle management, and the addition of simulation capabilities for effective revenue management. All of these things are made possible by the fact that it is a cloud-based solution.

Dynamic Offer Management

SunTec Dynamic Offer Management lets companies use advanced customer behaviour insights to adopt the "one-to-one" or "segment of one" strategy and quickly design and launch highly personalised product bundles and offers for any customer segment.

Deal Management

SunTec Deal Management lets companies automate and improve the whole deal management process, from deciding which products to buy to negotiating, making proposals, closing deals, and keeping track of commitments

.**Relationship Based Pricing Management**

SunTec is based on relationships Pricing Management lets companies divide customers into groups based on how they act and come up with new, more relevant prices for each group. It is easy to connect to existing transaction systems, automates the whole pricing process from start to finish, and gives you a central place to store and manage products and prices across the enterprise.

Enterprise Billing and Statements Management

SunTec Enterprise Billing and Statements Management makes it possible for companies to automate the whole billing process. This includes pricing, consolidation, pre-billing, billing, and tasks after billing, like settlement, dispute management, and arrears management.

Ecosystem Management and Monetization

SunTec Ecosystem Management and Monetization lets companies build their own ecosystems and make money from them or join forces with the ecosystems of other companies.

Benefits and Loyalty Management

SunTec Benefits and Loyalty Management lets businesses use a total relationship loyalty management strategy that encourages customers to spend more money, use more products and services, and sign up for more products and services across all business lines.

Enterprise Indirect Taxation Management

SunTec Enterprise Indirect Taxation Management lets organisations automate their indirect tax management programmes for compliance and efficiency, as well as handle the tax compliance requirements of multiple indirect tax regimes, such as GST and VAT.

CHAPTER 2

LITERATURE SURVEY

This section discusses numerous research of technologies used for Java upgrader.

2.1 MICROSERVICES

A business domain is broken up into a number of smaller, more manageable contexts when using a microservice architecture. These contexts are managed by a set of services that are autonomous, self-contained, only weakly coupled, and may be installed on their own. Netflix was one of the first firms to use them into their business. In 2009, far before the term "microservice" was ever coined, it began transitioning away from its formerly monolithic architectural design. Applications that are built on microservices expand well horizontally, both from a technical standpoint and in terms of how the organisation builds up development teams, which may be maintained small and flexible. This is true from both a technological and an organisational perspective. Adaptive business process management is already hard at work to make it such that these different sorts of options may coexist peacefully. In addition, providing the next degree of independence to agile teams via the separation of huge applications into individual microservices is beneficial to the development of agile methodologies.

Large, difficult applications may be deployed rapidly, often, and reliably with the help of an architecture called microservices. In addition to this, it enables the organization's technological stack to evolve over time. Using a microservices design allows for individual app components to function independently as their own services. Because their user interfaces and application programming interfaces (APIs) have been carefully crafted and are not too complex, these services are able to communicate with one another. Companies rely on services to assist them carry out their work, and each service is specialised to perform a certain function. Because

each service is capable of operating independently, it is possible to update, deploy, and scale it to fulfil the requirements of various components of an application. Each team could work on a distinct microservice and compose user stories that are relevant exclusively to the microservice that they are working on. [1].

2.2 SPRING BOOT

The popular Java-based Spring Boot framework for making web and enterprise apps, and how it gives service-oriented architecture the flexibility it needs (SOA). It can be hard to set up Spring-based apps because it's not always clear how they're set up. Spring Boot makes it easy to make stand-alone Spring apps that are ready for production and can be deployed with very little Spring configuration. Spring Boot makes it easier to manage dependencies by using a complete but flexible framework and all the libraries that go with it as a single dependency. This gives you all the Spring tools you need to start a project, which is more than you need for CRUD web apps. This framework has some extra features that are useful for many projects, like an embedded server, security, metrics, health checks, and configuration that can be done outside of the framework.

Spring Boot makes it easy to use microservice architecture because it sets itself up, includes servers, and makes it easy to manage dependencies. Setting up Spring Boot and getting the right application servers or packages doesn't take long. This makes it easier for development teams to create services quickly. Developers can "just run" the app, to use a phrase from Pivotal. Most development teams are already using, about to use, or planning to use a microservice architecture (80 percent based on our survey). Because of this and the fact that Spring Boot's main features are well-made to support and take advantage of a microservice architecture, Spring Boot has become an important part of the enterprise Java ecosystem and doesn't seem to be going anywhere soon. Most of the time, a war file is used to put a web app on a web server. Spring Boot apps, on the other hand, can be made as either a war file or a jar file. This means they can run without being set up or installed on the application server. [2].

2.3 MONGODB ATLAS

Document-oriented database systems, which are frequently referred to as "document stores," are helpful for the construction of current online applications that need rapid construction and deployment. Because the vast majority of data is kept in document-based formats such as JSON and XML, this is the case. It is simple to set up, manage, and grow in the cloud applications that use MongoDB Atlas since it is the hosted version of MongoDB as a Service. Users of MongoDB Atlas have the flexibility to exchange transient differences between replicas for reduced latency and increased availability during partitioning, much as users of many other NoSQL stores. In this article, we discuss a research that we conducted out in the real world to determine what percentage of the data included in MongoDB Atlas is inaccurate. It is simple to set up and operate databases whenever and wherever they are required with the assistance of MongoDB Atlas. A device on a network is assigned a number known as its IP address, which other devices on the network use to identify it. We are only able to connect to a cluster using Atlas when we are using an IP address that we are certain is secure. A "Whitelist" is a list of trustworthy IP addresses that may be used to connect to our cluster and obtain access to our data. This list can be created in Atlas, and it is one of the features that it offers.

The term "cloud database" refers to the service that is used by cutting-edge applications all over the globe. The automation and practises that are considered to be among the best in their respective fields ensure that the system is constantly available, has the capacity to expand as required, and complies with the most stringent requirements for data security and privacy. We are able to move things along more quickly and spend less time maintaining our database because to the many drivers, connectors, and tools that come included with MongoDB. [3].

2.4 MAVEN

Maven is a great tool for running projects. It has to do with POM (project object model). It is used to plan, build, and keep track of projects. Maven makes it easier for Java programmers to do their jobs every day and makes it easier to understand any Java-based project. In Maven

repositories, you can find lists of JAR files and information about them. Metadata are POM files that describe the projects that each packaged JAR file is a part of, including what external dependencies each packaged JAR file needs. Using this information, Maven can download the dependencies of your dependencies until all of the dependencies are on your local machine. Building a project with Maven takes six steps, and each step depends on the one before it.

These steps make up Maven's default lifecycle. - Validate checks to see if the project has all the settings it needs to be built. Compile turns the project's source code into the raw deliverables for the project (e.g., .class files). To find bugs, test builds and runs test suites. When you package something, it's ready to use (e.g., .jar, .war, .ear). Install puts the deliverables where they belong on the local system. When you click "Deploy," the local installation is sent to where it will be used. Maven is a tool that helps developers manage and understand projects by giving them a complete framework for the build lifecycle. Maven has a standard directory structure and standard build lifecycle, so it doesn't take long for the development team to automate the project's build infrastructure.

Maven can set up the way to work so that it follows standards in a setting with many development teams in a very short amount of time. Maven makes it easy for developers to set up automated reports, checks, builds, and tests. [4].

2.5 POSTMAN

A test in Postman is just a piece of JavaScript code that is run after a request has been sent and a response has been received from the server. It's very easy to use POSTMAN. It gives a list of API calls, and to test the APIs of an app, you have to follow that list. For each request, Postman lets you write and run tests. JavaScript is the language used to write these tests. Download the app Postman to your phone. It can be downloaded as a native app for macOS, Windows, and Linux. There's also a Chrome app for it. If you use Chrome, click on the square that says "Apps." The most common ones are GET (to get data), POST (to change data in an existing file), PUT (to replace an existing file), and DELETE (to get rid of data) (Delete data from server). Send Request: a. In the menu bar at the top of the page, click "New." b. Type the

request URL (URL-REST endpoint) as shown in the image below. URL: http://get.postman-echo.com. Users can save requests for later use by clicking the Save button next to the Send button. You can link up to the tool that helps you manage Build with Newman. It is a command line tool that works with Postman and lets you connect Postman to a build management tool to help automate testing. Authorization: The authorization process checks to see if you are allowed to get the information you want from the server. It makes sure that safety is there. We'll talk about the different permissions that can be given to the tool in the next part of this paper. [5].

2.6 API GATEWAY

A new term in software architecture patterns is "micro-service approach." It is becoming more popular because it is flexible, uses a granular approach, and has services that are only loosely linked. [I]. In the first part of this paper, we talk about an API Gateway System, which is the gateway to backend services. It can make it easier for internal services to talk to each other and cut down on the number of remote calls between apps and back-end services. Second, this paper makes a Kubernetes and Prometheus-based auto-scaling system for the API Gateway System that can change the number of application instances on its own. It can make better use of the system's resources while making sure the application's service is always available and of high quality. The micro-service is a new term in software architecture patterns that divides a complicated system into a number of small, separate services. Each service has its own process that does a specific job and runs on its own. Services can talk to each other using light protocols like HTTP, RPC, and so on. In the micro-service architecture, methods called from inside a process become remote procedure calls. In the past, applications would directly call back-end services. It is the easiest way to get the job done. On the back end, though, different services could be called from an app. The delays could be caused by a lot of calls from far away, which could make customers feel bad. As a business grows, it may need to split the backend service into a number of small, separate services. The direct communication pattern could mean that all applications will have to change a lot to keep up with changes to services. By putting the API Gateway System between applications and back-end services, it can hide some limits and details from applications. For example, applications don't know when the service on the back end changes. But since the only way to get to back-end services is through

the API Gateway System, if it stops working, all services could stop working. So, API Gateway services need to be up and running most of the time. [6] .

2.7 JAVA-TO-JAVA (J2J) SOURCE CODE MIGRATOR

Scaled-down versions of the regular Java Virtual Machine (JVM) are often used in embedded systems that run on JVM-based processors. These versions do not support the full breadth of Java byte codes and native methods that one would expect a JVM to be able to make use of. As a result, the code bases, such as Java libraries, need to be migrated in order for them to be capable of being run on the embedded processor that is based on JVM. This article describes Monarch, a Java-to-Java (J2j) source code migrator that we are building to aid with code migrations and that is mentioned in this post. We are doing this since this article features Monarch. [7].

CHAPTER 3
METHODOLOGY

3.1 EXISTING SYSTEM

Developers have to update the file when they switch from one version of Java to another platform. There are a lot of pom files for each project. Under the system we have now, each file must be changed separately. To change each file one by one would take days. Because of this, it takes a lot of time to use the current system.

3.2 PROPOSED SYSTEM

Compared to other models, the proposed Java Upgrader has done very well in terms of accuracy and speed. The new feature of the model is that it makes it easy to switch between versions of Java. There are a lot of pom files for each project. This tool automatically changes all of the old content to new content, so you don't have to update each file individually. With the tool that was suggested, we can quickly switch from one version of a Java project to another.

3.3 WORKING OF MODULES

For this tool, we have two services for the upgrade and configuration for the migration.

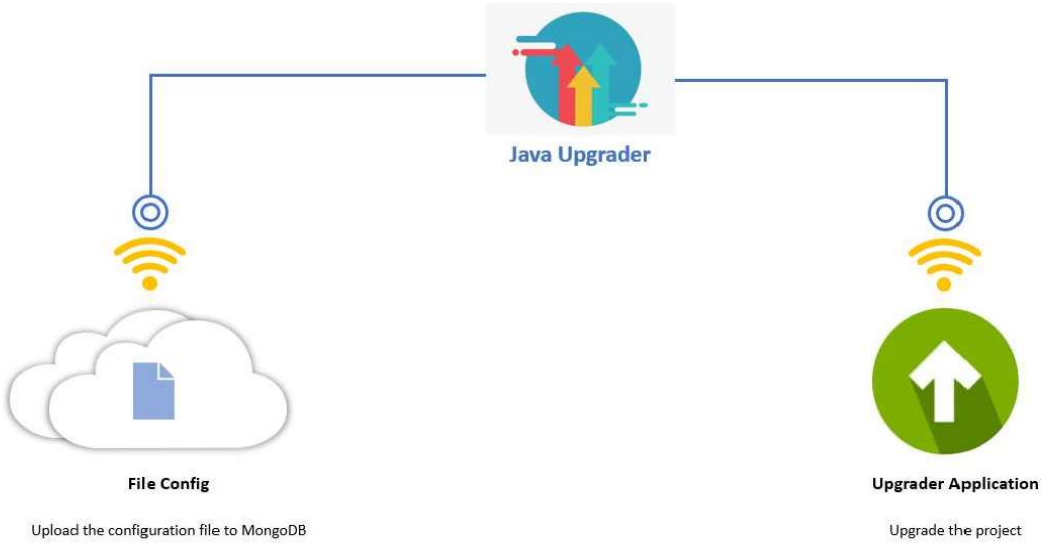


Figure 1: Working Modules of Java Upgrader

3.3.1 File Config

This service is used to upload configuration file to MongoDB Atlas. The configuration file, which contains all Java plugins, JDK version etc.

Function included in the services are:

3.3.1.a Upload – Upload file to the DataBase

3.3.1.b Delete – Delete all the collection inside the MongoDB Atlas

3.3.1.c Local Upload – Upload the configuration file inside the project itself if the DB is down

3.3.2 Upgrader Application

It will upgrade the pom.xml and java files according to the configuration file. It will retrieve content from configuration file in the MongoDB Atlas.

Function included in the service:

- Search File – It will search all the pom and Java files whichare mentioned in the configuration file.
- Json To HashMap - It will convert json content to Hashmap
- Modify Files– It will replace the old content with new content in the configuration file. It will replace according to the parent node.

3.4 ARCHITECTURE

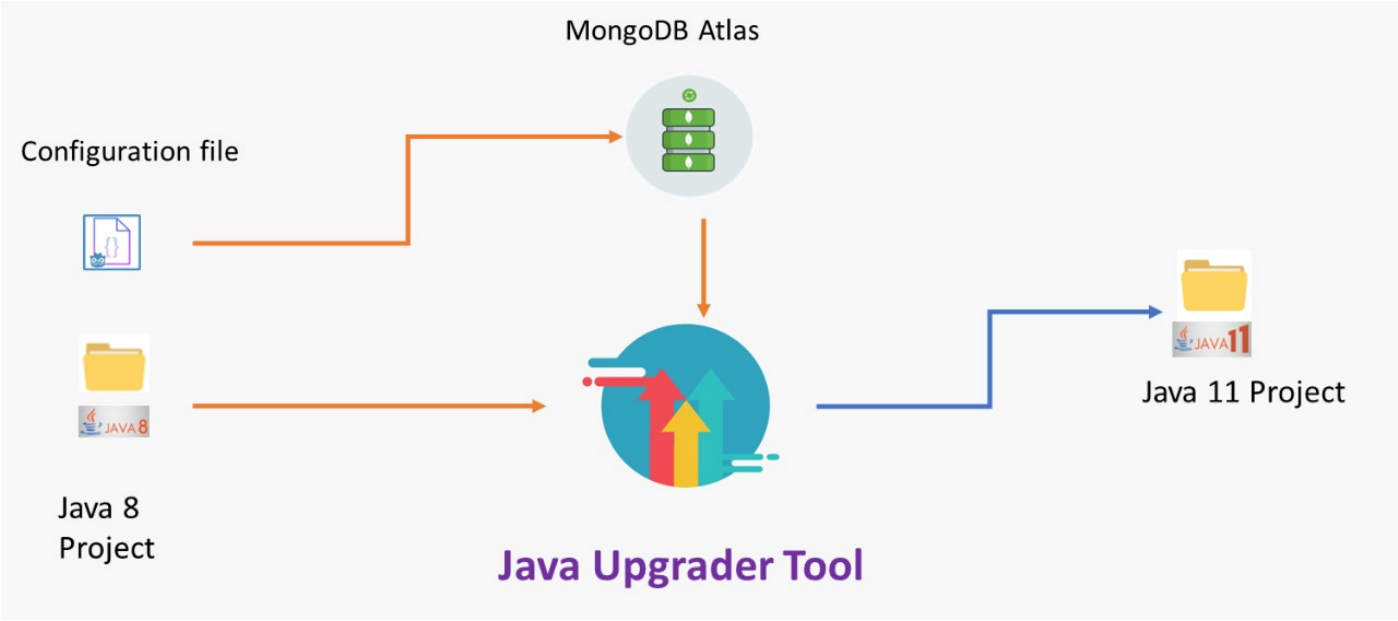


Figure 2: Architecture of Java Upgrader Tool.

3.4.1 Configuration.json file

The configuration.json file contains the path, parent nodes, old value, and new value. This is where our upgrade information is collected.

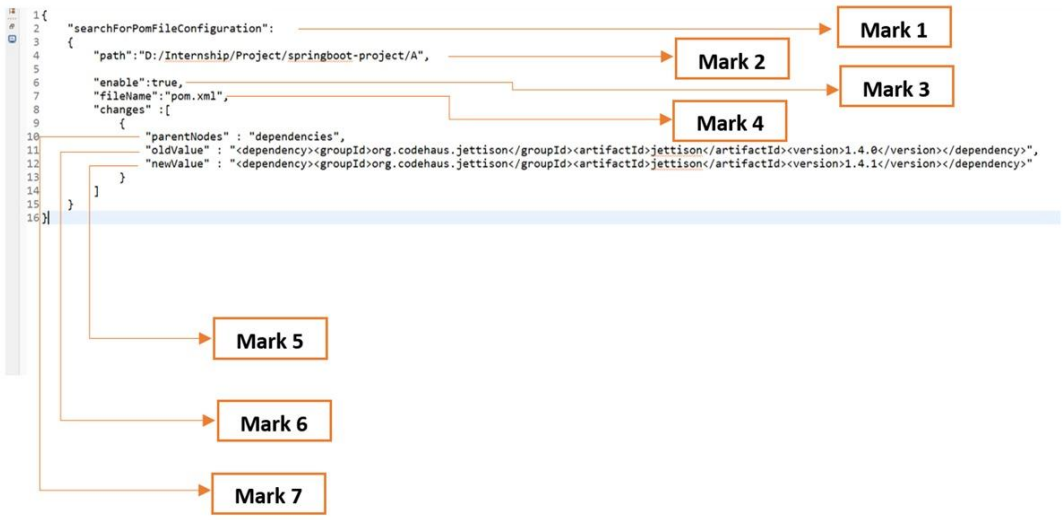


Figure 3: Configuration of JSON file

Mark 1: - Name of the configuration.

Mark 2: - Specifies the path to the directory where all the file needs to be updated.

Mark 3: -Indicate whether the configuration is enabled or disabled.

Mark 4: - Specify the file that needs to be changed.

Mark 5: - Specify the new dependency/new code in the files that need to be removed.

Mark 6: -Specify the old dependency/old code in the files that need to be appended.

Mark 7: - When the entire child node needs to be change you have to specify the parent node.

This tool not only upgrades the JDK, it also does some other functionality.

Special Cases

- Case 1

  Sometimes, the configuration file doesn't list the names of the files. In this case, the tool

will find the files that have the old values on their own.

- Case 2

  If only one line of code in one of the pom files or other files changes, the parent node is not needed.

- Case 3

  When a child node needs to be changed, the parent node must be specified.

- Case 4

  If you want to delete a line (the old value in the configuration) from a file, leave the new value blank.

- Case 5

  To add a new line to a specific line in a file, just join the old value and the new content with /n and set that as the new value in the configuration file.

3.4.2 Microservices

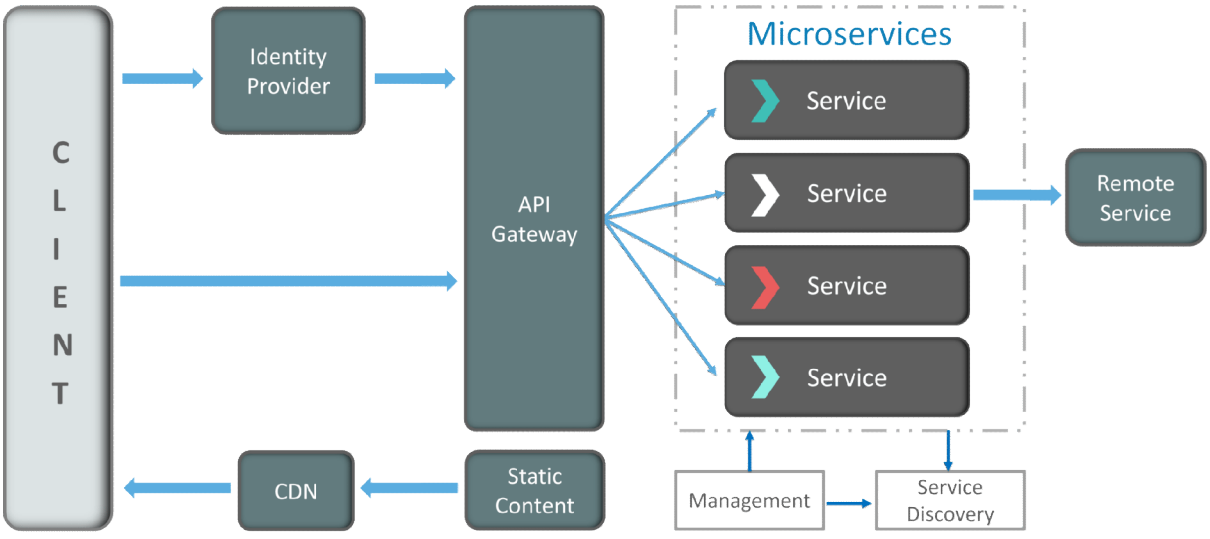


Figure 4: Architecture of Mircoservices

With Java Upgrader, an application is set up as a group of services using the microservice

architecture. We can find errors so quickly with this architecture because we can see which service instances are running.

The services used in this project are as follows: -

- Cloud Gateway

  A cloud gateway is a single point of entry and exit for many microservices, like File config and Upgrader. We can also have more than one service in the project use the same port number. Also, all of the project's services can use the same port number.

- Cloud Config Server

  In the version controller, the same code will be set up for all property files (Git). This code can be changed by any service that has access to the git repository. We can shorten the code by doing this. Also, if you change the common property file, you don't have to change each file separately. Instead, you should change the code in the repository.

- Service Registry

  A service registry is a database that keeps track of how many copies of each application's microservice are available. Eureka Server is an app that keeps track of the information for each client-service app. Each Micro service will sign up with the Eureka server, and the Eureka server will know which client apps are running on each port and IP address. Eureka Server is also called Discover Server. The Spring Cloud package includes Eureka Server. To do this, we need to build the Eureka server and run it on port 8761, which is the default.

- File Config

  With this service, you will send the configuration file to the Mongo DB Atlas using a Postman operation. Adding a file to the Mongo DB Atlas is one thing you can do. Taking a collection out of the Atlas is another option..

- Upgrader

  This service will get the JSON data from the mongo DB Atlas configuration file. Thejson content is read into this service from a configuration file in the MongoDB Atlas. It will change the old value to the new value based on the configuration file. It will also find the exact file shown in the configuration file by its extension. It will also look for the file with the extension listed in the configuration file..

3.5 TECHNOLOGIES USED

3.5.1 SpringBoot

Java Spring Framework, also called Spring Framework, is a popular open source enterprise-levelframework for making apps that run on the Java Virtual Machine that can run on their own (JVM). Java Spring Boot, also known as Spring Boot, is a tool that makes building web apps and microservices with Spring Framework faster and easier. This lets programmers make modular apps with loosely connected parts that work well for microservices and distributed network apps.Spring Boot is a free Java web framework that can be used to build websites. It is made up of many small services. Using the pre-built code in its codebase, the Spring Boot framework creates an environment that is ready for production and can be changed in any way. With the microservice architecture, developers can make an app that is completely closed off and has application servers built right in.

Spring Boot makes it easy to use microservice architecture because it sets itself up, includes servers, and makes it easy to manage dependencies. Setting up Spring Boot and getting the right application servers or packages doesn't take long. This makes it easier for development teams to create services quickly. Developers can "just run" the app, to use a phrase from Pivotal. Most development teams are already using, about to use, or planning to use a microservice architecture(80 percent based on our survey). Because of this and the fact that Spring Boot's main features are well-made to support and take advantage of a microservice architecture, Spring Boot has become an important part of the enterprise Java ecosystem and doesn't seem to be going anywhere soon.

3.5.2 MongoDB Atlas

MongoDB Atlas is a database service for more than one cloud that is made by the same people who make MongoDB. Atlas makes it easier to set up and manage your databases, and it gives you the freedom you need to build fast, reliable global apps on the cloud service of your choice.

With MongoDB Atlas, it's easy to set up and run databases whenever and wherever they're needed.

The IP address of a device is a number that tells other devices on the same network what that device is. We can only connect to a cluster from a safe IP address when we use Atlas. Inside Atlas, we can make a "Whitelist," which is a list of trusted IP addresses that can be used to connect to our cluster and get to our data.

Modern apps all over the world use this service to store data in the cloud. Best-in-class automation and tried-and-true practises make sure that the system is available, scalable, and meets the strictest data security and privacy standards. We can get things done faster and spend less time managing our database because MongoDB has a lot of drivers, integrations, and tools.

3.5.3 Microservices

Microservices, also called "microservice architecture," are a type of architecture that builds an app out of a group of services that work together.

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

With the microservice architecture, large, complicated applications can be delivered quickly, often, and reliably. It also lets the technology stack of an organisation change over time. With a microservices architecture, different parts of an app work as separate services. These services can talk to each other because their interfaces and application programming interfaces (APIs) arewell-defined and not too heavy. Businesses use services to help them do their jobs, and each service does only one thing. Each service can run on its own, so it can be updated, deployed, and scaled to meet the needs of different parts of an application.

- Service Registry

  A service registry is a database that keeps track of how many copies of each application's microservice are available. When a service goes offline or is no longer available, or when a new service comes online, the service registry needs to be changed. You can do this on your own or hire someone else to do it..

- Discovering Services

  A client or API gateway needs to know where a service is before it can send an API request to it. Since the microservice architecture is set up so that each service can have multiple instances, load balancing and service discovery go hand in hand. Note that load balancing requests across service instances is different from load balancing incoming client traffic across multiple API gateway nodes..

- API Gateway

  With microservices architecture, we can put different services on different servers in a private network. These servers are called "hosts." Before microservices handle a request from a client, they should make sure it is valid. An API Gateway is a single point where applications or clients from the outside can connect to and disconnect from multiple microservices. It is also called Edge Microservice. Since clients from outside the company can't access the microservices directly, it acts as a go-between for them and the group of microservices. In real life, an external client could be a desktop programme, a mobile app, a third-party programme, a service from somewhere else, etc..

- Resilience4j

  When we build an app, especially one that uses Microservices, there is a good chance that it will change when we run it in real time. It could be a slow response, a broken network, a failed REST call, a failure caused by too many requests, or a lot of other things. Our application needs a Fault Tolerance system so that it can handle these kinds of possible errors. We'll do it with the help of the Resilience4j library. Resilience4j is a small, easy-to-use library that keeps working even if something goes wrong. It was based on Hystrix on Netflix, but it was made for Java 8 and functional programming..

- Zipkin

  With Zipkin, you can find people who are in different places. It helps get the timing information needed to fix problems with latency in service architectures. This information is taken in and looked up as part of the features. In a log file, you can go straight to a trace ID. You can search by service, operation name, tags, duration, and other things if you don't want to do that. You will get a summary of some interesting information, like how much time was spent in a service and whether or not operations failed. A Dependency diagram is also part of the user interface for Zipkin. This picture shows how many tracked requests each application handled. This can help find things like error paths or calls to services that are no longer supported.

3.5.4 Maven

Maven is a great project management tool. It's related to POM (project object model). It's used toplan projects, build them, and keep track of them. Maven makes it easier for Java programmers to do their jobs every day and makes any Java-based project easier to understand. You can find lists of JAR files and information about them in Maven repositories. Metadata are POM files that describe the projects that each packaged JAR file is part of, including what external dependencies each packaged JAR file needs. With this information, Maven can download the dependencies of your dependencies until all of the dependencies are on your local machine. There are six steps to building a project with Maven, and each step depends on the one before it. The default lifecycle of Maven is made up of these steps. - Validate checks to see if the project has all of the settings it needs to be built. Compile takes the project's source code and turns it intothe project's raw deliverables (e.g., .class files). Test builds and runs test suites to find bugs. When you put something in a package, it's ready to be used (e.g., .jar, .war, .ear). Install puts the deliverables on the local system where they belong. When you click "Deploy," the local installation is sent to the place where it will be used. Maven is a tool that helps developers manage and understand projects by giving them a full framework for the build lifecycle. Maven has a standard directory structure and a standard build lifecycle, so it doesn't take long for the.

development team to automate the project's build infrastructure. In a setting with many development teams, Maven can set up the way to work so that it follows standards in a very short amount of time. Developers can easily set up automated reports, checks, builds, and tests with Maven.

3.5.5 Postman

Postman is where you can make and use APIs. It is a place where APIs can run. Postman helps you make better APIs faster by making each step of the API lifecycle easier and making it easier for people to work together. Postman Enterprise is made for companies that want to use Postman on a large scale and need support, security, reliability, and uptime on an enterprise level. With these features and benefits, the Postman API Platform helps people get a lot done.

Features

- Postman as the single source of truth for all APIs
- Up-to-date and easily discoverable API documentation
- Instant search across your API landscape
- Persona-based pre-configured API workspaces

Benefits

- Eliminate duplicate work
- Onboard developers faster
- Fewer meetings = happier developers

Connect Postman to an identity provider so that users can safely log in to Postman Enterprise. Postman Enterprise works with Okta, OneLogin, Duo, Ping Identity, AD FS, GSuite, and even customSAML.

CHAPTER 4
RESULT AND DISCUSSION

In the proposed Java Upgrader Tool, will perform very well in terms of accuracy and time when compared to the existing models. The novelty of the proposed model is that it automates the update from one Java version to another. Each project contains hundreds of pom files. Rather than updating individual files, this tool replaces all of the old content with new content automatically.Using the proposed tool, we can upgrade a Java project from one version to another in just a few seconds.

4.1 TESTING METHODS

There are lots of ways to test something. Testing is used in this project to find errors in the requirements, design, or code.

4.1.1 Unit Testing With Mockito

Unit testing is a part of how software is made. During this step, "units," which are the smallest parts of an app that can be tested on their own, are checked to make sure they work correctly. Unit testing is an important part of the development process because, if done right, it can help find bugs in the code early on, when they might be harder to find. Test-driven development (TDD) is a way to build a product carefully by testing it over and over and making changes. Unit testing is part of TDD. This type of testing is also the first level of testing software. It is done before other testing methods, like integration testing. Unit tests are usually run by themselves to make sure that a unit doesn't depend on any code or functions from outside the unit. Testing can be done by hand, but most of the time it is done automatically. Mockito is a well-known open source framework for testing software by simulating objects. Using Mockito makes it much easier to make tests for classes that depend on things outside of the programme. A fake copy of a class or interface is called a "mock object." It lets you choose what will happen when a method is

called. Most of the time, tests show how they track how people use the system. The main parts of a unit test are the plan, the cases and scripts, and the actual unit test. The first step is to get ready for the unit test and review it. The next step is to create test cases and scripts, and then the code needs to be tested. For test-driven development to work, developers must first create unit tests that fail. Then they write code and change the app until the test passes. Most of the time, TDD results in code that is clear and easy to understand.

4.1.2 Integration Testing

Integration testing is the next step in the testing process. During this testing, groups of software units or parts are checked at the same time. At the integration testing level, the goal is to find bugs that happen when two or more parts or units that have been tested separately work together. Integration testing looks at how well all the modules work together. During unit testing, each module is tested on its own. The software is made up of different software modules that were coded by different coders or programmers. Integration testing makes sure that all of the modules can talk to each other in the right way.

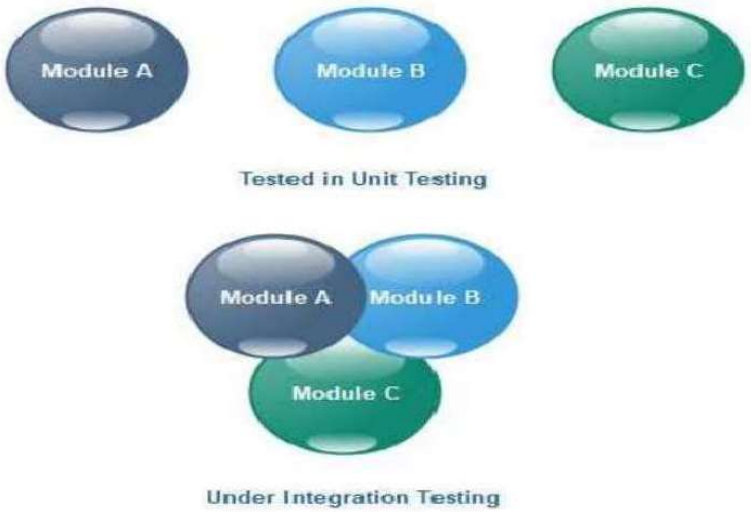


Figure 5: Integration Testing

4.2 OUTPUTS

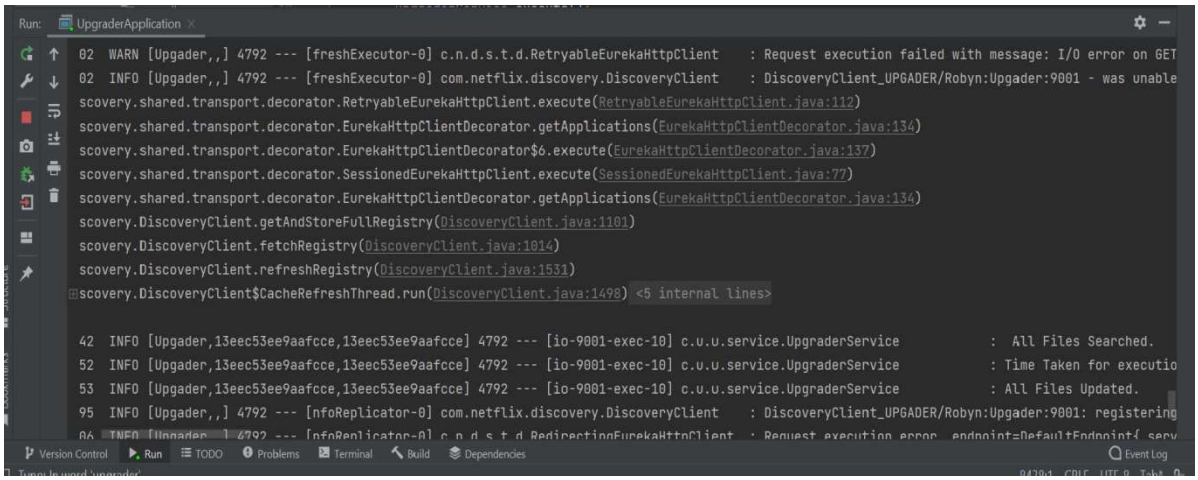


Figure 6 : Log for upgrader services while running

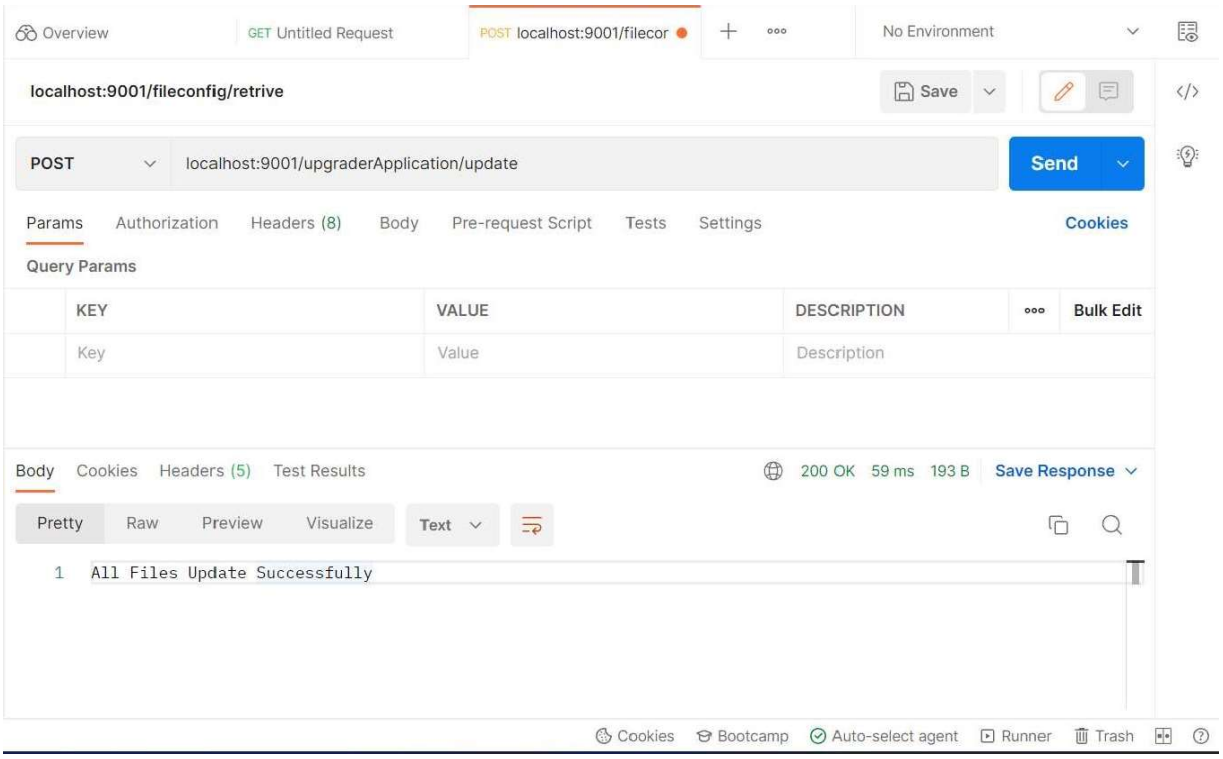


Figure 7 : API for updating pom.xml files in postman

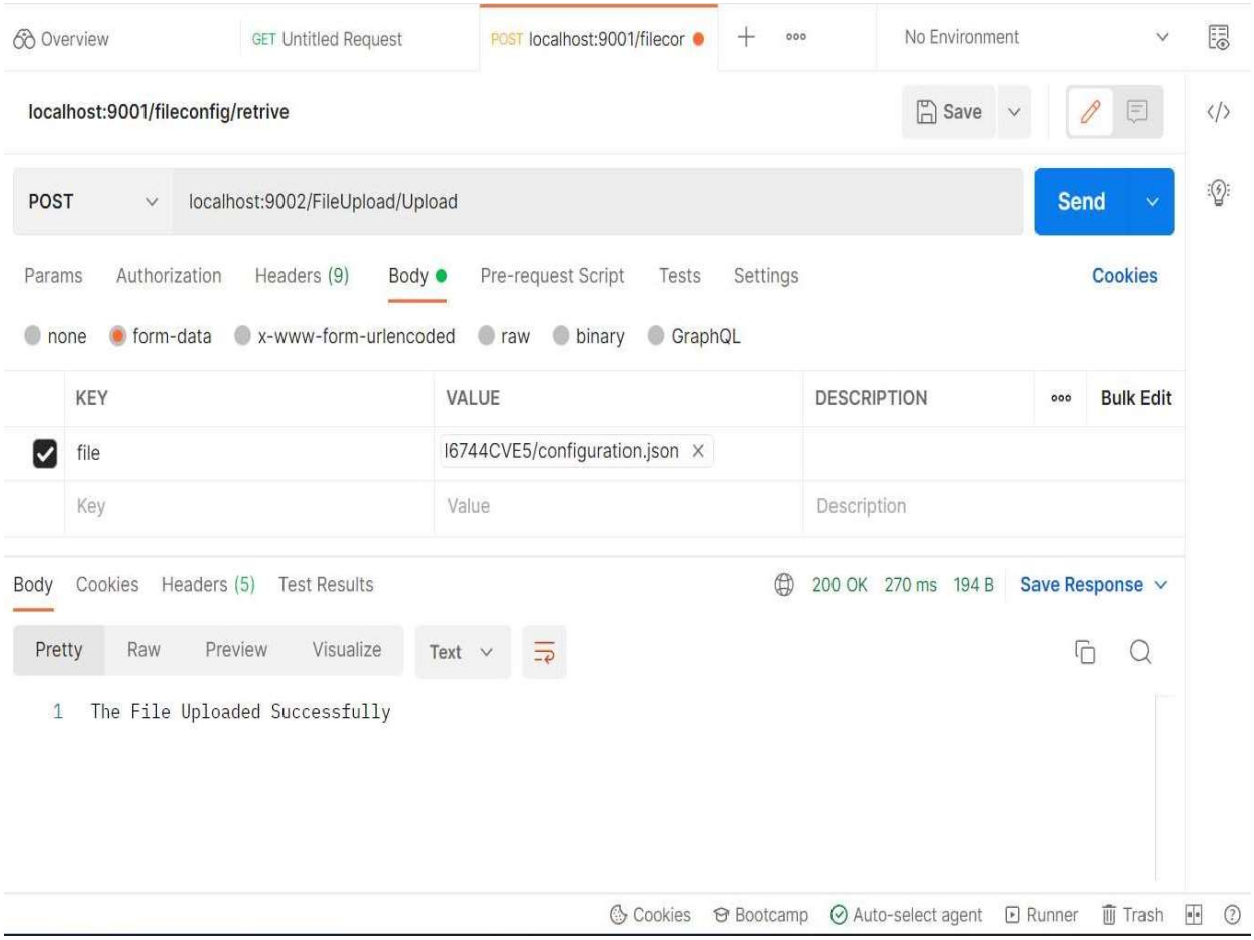
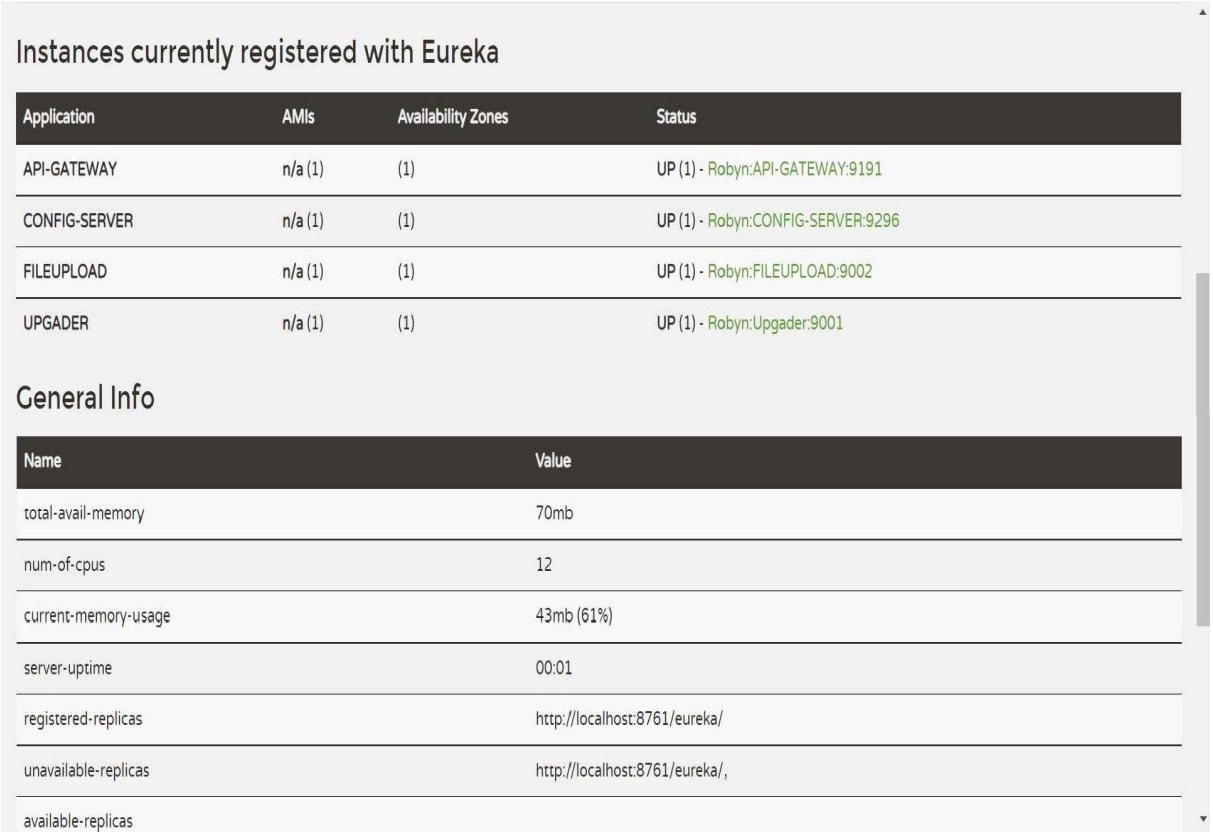


Figure 8: API for uploading Configuration file to MongoDB Atlas via Postman



Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - Robyn:API-GATEWAY:9191
CONFIG-SERVER	n/a (1)	(1)	UP (1) - Robyn:CONFIG-SERVER:9296
FILEUPLOAD	n/a (1)	(1)	UP (1) - Robyn:FILEUPLOAD:9002
UPGADER	n/a (1)	(1)	UP (1) - Robyn:Upgader:9001

General Info

Name	Value
total-avail-memory	70mb
num-of-cpus	12
current-memory-usage	43mb (61%)
server-uptime	00:01
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/,
available-replicas	

Figure 9 : Instance currently registered with Eureka

CHAPTER 5
CONCLUSION

For most applications migrating from one LTS to the next LTS takes a couple of hours to a couple of days. Most of the time is spent building the application. It's important to get started and make incremental changes. This tool will automate the process of upgrading from one Java version to another. Some of the projects started some years ago were implemented using an older Java version (LTS) that was the most stable version of the language and virtual machine at that time. However, as time went by, new Java (LTS) took the lead as the official, latest LTS version, so we need to start migrating our projects to that version. This is a project which is basically use for change the old value to new value for all files within a project. This project reads the old content and replace with the new content, this works like a changing of versions from old to new version e.g., JDK1.8 to JDK11. This will also change the libraries which is use for import in a program. As well as upgrading the JDK, this tool also modifies files that we need to mention in the configuration file we send to the tool. The proposed Java Upgrader has performed very well in terms of accuracy and time when compared to the existing models

5.1 FUTURE ENHANCEMENT

There are some limitations to the proposed upgrader tool. We need a configuration file that is not auto-configured for the migration. The developer should manually configure the file according to the JDK we need to upgrade. When manually configuring files, it is necessary to research which dependencies are supported and which are compatible. As a result, creating a configuration file takes a little while.

These are some problems with the system that is being proposed. In the future, artificial intelligence will be able to solve these kinds of problems. With the help of AI, the configuration file for the JDK we want to switch to can be set up automatically.

REFERENCES

[1]. Grzegorz Blinowsk, Anna Ojdowska, and Adam Przybyłek, “Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation”, Institute of Computer Science,Warsaw Universityof Technology IEEE 2022, 00-665 Warsaw, Poland

[2].Kavya Guntupally ,”Spring Boot based REST API to Improve Data Quality Report Generation for Big Scientific Data: ARM Data Center Example”, 2021 IEEE International Conference on Big Data (Big Data)

[3] Huang, C., Cahill, M., Fekete, A., Röhm, U,” Data Consistency Properties of Document Store as a Service (DSaaS): Using MongoDB Atlas “,Lecture Notes in Computer Science(), vol 11135. Springer, 2019

[4].Casimir Désarmeaux, Andrea Pecatikov, and Shane McIntosh.”The Dispersion of Build Maintenance Activity across Maven Lifecycle Phases”.2021 IEEE/ACM 13th Working Conference on Mining Software Repositories

[5].Dheeraj , Kalpana Sharma, “Security Testing of API using Postman and Swagger tools andits use in Internet of Things (IOT)”,2020

[6] Meina Song, Chengcheng Zhang, E Haihong, "An Auto Scaling System for API Gateway Based on Kubernetes", 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)

[7] Victor L. Winter, Jonathan Guerrero, Carl Reinke,” Monarch: A High-Assurance Java-to-Java (J2j) Source-Code Migrator”, 2011 IEEE 13th International Symposium on High-Assurance Systems Engineering

[8] J. Martin and H. A. Müller.,” Strategies for Migration from C to Java.”, In Proceedings of the 5th European Conference for Software Maintenance and Reengineering, Lisbon,

[9]I. K. Aksakalli, T. Celik, A. B. Can, and B. Tekinerdogan, ‘‘Systematic approach for generation of feasible deployment alternatives for microservices,’’ IEEE Access, vol. 9, pp.

29505–29529, 2021.

[10] G. Mazlami, J. Cito, and P. Leitner, ‘‘Extraction of microservices from monolithic software architectures,’’ in Proc. IEEE Int. Conf. Web Services (ICWS), Jun. 2017, pp. 524–531, doi: 10.1109/ICWS.2017.61.