**Library Management System Using Jenkins CI/CD pipeline**

**A Project Report**

**Submitted in partial fulfilment for the award of the degree Of**

**Bachelor of Technology**

**In**

**Computer Science and Engineering**

**By**

## Chikkudu Chandu (22261A0512)

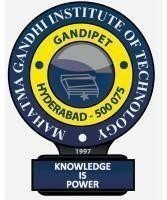
**Naika Navaneeth (22261A0535)**

**Under the guidance of**

**Dr K. RAJITHA**

**ASSISTANT PROFESSOR**

**Department of Computer Science & Engineering**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

Kokapet(Village), Gandipet, Hyderabad, Telangana – 500075. [www.mgit.ac.in](http://www.mgit.ac.in/)



We **Chikkudu Chandu** and **Naika Navaneeth**, bearing the roll numbers, **22261A0512** and **22261A0535**, hereby declare that the project entitled Library Management System Using Jenkins CI/CD pipeline under the guidance of **Dr K. Rajitha**, **ASSISTANT PROFESSOR**, Department of Computer Science and Engineering, MGIT, Hyderabad, is submitted in the partial fulfilment of the requirement for qualifying the project work course during the academic year 2024-2025.

Project Coordinator Head of the Department

Dr K. Rajitha Dr. C.R.K Reddy

Dept. of CSE Dept. of CSE

# ACKNOWLEDGEMENT

I extend my sincere and heartfelt thanks to our esteemed faculty, Dr K. Rajitha, Assistant Professor and for his exemplary guidance, monitoring and constant encouragement throughout the course at crucial junctures and for showing us the right way.

I am grateful to respected coordinator Dr K. Rajitha, Assistant Professor for permitting me to utilize all the necessary facilities of the Institute.

I would like to extend thanks to our respected Head of the department, Dr C.R.K Reddy for allowing us to use the facilities available. We would like to thank other faculty members also.

Last but not the least, I would like to thank our friends and family for the support and encouragement they have given us during the course of our work.

CHIKKUDU CHANDU (22261A0512)

NAIKA NAVANEETH (22261A0535)

# ABSTRACT

This research focuses on the implementation of a CI/CD pipeline using Jenkins, showcasing its importance in modern DevOps practices. Continuous Integration and Continuous Deployment (CI/CD) play a pivotal role in automating the software development lifecycle, enabling seamless code integration, testing, and deployment. Jenkins, an open-source automation server, is utilized to streamline these processes, ensuring efficient collaboration and faster delivery of high-quality software.

The paper explores the key aspects of setting up a CI/CD pipeline, including version control with GitHub, automated build processes, unit testing using JUnit, and deployment automation. These principles are demonstrated through a practical application: the Library Management System. The system automates essential library operations, such as book management, user tracking, and transaction handling, providing a reliable and scalable solution.

This research highlights how DevOps practices, supported by CI/CD pipelines, improve productivity, reduce errors, and ensure consistency in software development and deployment. It concludes by discussing the potential for expanding the system’s capabilities, leveraging the robust foundation provided by Jenkins and DevOps methodologies.

# TABLE OF CONTENTS

|  |  |
| --- | --- |
| **TITLE**  01.INTRODUCTION | **PAGE NO**  06 |
| 02.FLOW OF EXECUTION | 07 |
| 03.TECHNOLOGIES USED,CHALLENGES AND ADVANTAGES | 08 |
| 04.CODES | 09 |
| 05.OUTPUTS | 13 |
| 06.FUTURE SCOPE | 14 |
| 07.CONCLUSION | 15 |
| 06.REFERENCES | 16 |

**INTRODUCTION**

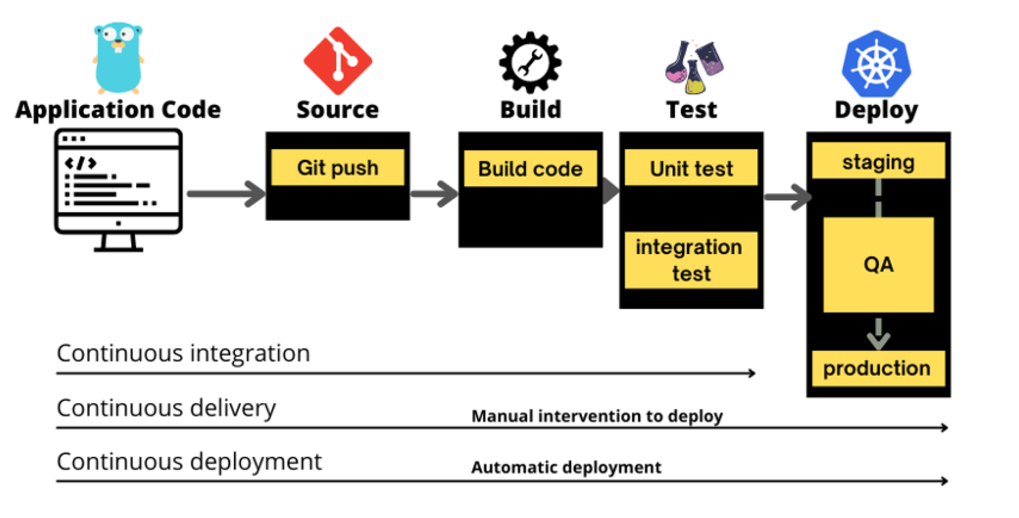
Continuous Integration and Continuous Deployment (CI/CD) pipelines have become a cornerstone of modern software development, enabling teams to deliver high-quality software faster and more reliably. In this project, we implemented a CI/CD pipeline using Jenkins to automate the build, test, and deployment processes. The pipeline ensures that every code change is tested and deployed seamlessly, fostering collaboration and rapid feedback.

Jenkins, an open-source automation server, plays a vital role in this project by orchestrating the entire CI/CD process. It pulls the latest code from the repository, compiles the code, runs automated tests, and deploys the application. This eliminates manual steps, reduces errors, and speeds up the development lifecycle.

Our project, the Library Management System, demonstrates these principles in action. It is designed to automate library operations, such as book management, user tracking, and transaction handling, providing an efficient and reliable solution for libraries of all sizes.

## FLOW OF EXECUTION

Our project followed these steps for execution:  
**1. Code Development**: Core classes (Book, Library, LibrarySystem) and test cases were written.  
**2. Version Control:** Code changes were managed using GitHub.  
**3. Pipeline Setup:** A Jenkins pipeline was created to automate the following:  
 - Checkout: Fetch the latest code from the GitHub repository.  
 - Build: Compile Java files to ensure the code is error-free.  
 - Test: Run JUnit test cases to validate functionality.  
 - Deploy: Package the application into a JAR file and deploy it to the target directory.  
**4. Feedback Loop:** Monitor pipeline logs to resolve issues and improve the system iteratively.



**Technologies used:**

* Programming Language: Java, for implementing the core logic of the system.
* Testing Framework: JUnit, for creating and running test cases.
* Continuous Integration/Delivery: Jenkins, for automating the build, test, and deployment process.
* Version Control: Git and GitHub, for managing the source code and tracking changes.
* Operating System: Windows, for development and execution.

**Challenges and Advantages:**

**Challenges:**  
1. Ensuring compatibility between Jenkins and the development environment.  
2. Debugging pipeline errors during initial setup.  
  
**Advantages:**  
1. Automated workflows reduced manual intervention, saving time and effort.  
2. Early detection of issues through automated testing improved the code quality.

CODE

1.Code used for CI/CD pipeline:

pipeline {

agent any

stages {

stage('Checkout Code') {

steps {

git url: 'https://github.com/chikkuduchandu/MyFirstJenkinsProject.git', branch: 'main'

}

}

stage('Build') {

steps {

echo 'Compiling Java files...'

bat 'javac HelloWorld.java'

}

}

stage('Test') {

steps {

echo 'Running tests...'

bat 'javac -cp ".;C:\\Users\\chand\\Documents\\V sem\\CSE\\CSE\\DEVOPS\\project\\junit-4.10.jar" HelloWorldTest.java'

bat 'java -cp ".;C:\\Users\\chand\\Documents\\V sem\\CSE\\CSE\\DEVOPS\\project\\junit-4.10.jar" org.junit.runner.JUnitCore HelloWorldTest'

}

}

stage('Deploy') {

steps {

echo 'Deploying application...'

bat 'jar cvf HelloWorld.jar HelloWorld.class'

bat 'xcopy /Y HelloWorld.jar "C:\\Users\\chand\\Documents\\V sem\\CSE\\CSE\\DEVOPS\\project\\"'

}

}

}

post {

always {

echo 'Cleaning up...'

}

}

}

2.Java codes for Library Management:

1. Book.java

public class Book {

    private String title;

    private String author;

    private boolean isCheckedOut;

    public Book(String title, String author) {

        this.title = title;

        this.author = author;

        this.isCheckedOut = false;

    }

    public String getTitle() {

        return title;

    }

    public String getAuthor() {

        return author;

    }

    public boolean isCheckedOut() {

        return isCheckedOut;

    }

    public void checkOut() {

        isCheckedOut = true;

    }

    public void returnBook() {

        isCheckedOut = false;

    }

    @Override

    public String toString() {

        return title + " by " + author + (isCheckedOut ? " (Checked out)" : "");

    }

}

1. Library.java

import java.util.ArrayList;

import java.util.List;

public class Library {

    private List<Book> books;

    public Library() {

        books = new ArrayList<>();

    }

    public void addBook(Book book) {

        books.add(book);

        System.out.println("Added: " + book);

    }

    public void viewAvailableBooks() {

        System.out.println("Available Books:");

        for (Book book : books) {

            if (!book.isCheckedOut()) {

                System.out.println(book);

            }

        }

    }

    public void checkOutBook(String title) {

        for (Book book : books) {

            if (book.getTitle().equalsIgnoreCase(title) && !book.isCheckedOut()) {

                book.checkOut();

                System.out.println("Checked out: " + book);

                return;

            }

        }

        System.out.println("Book not found or already checked out.");

    }

    public void returnBook(String title) {

        for (Book book : books) {

            if (book.getTitle().equalsIgnoreCase(title) && book.isCheckedOut()) {

                book.returnBook();

                System.out.println("Returned: " + book);

                return;

            }

        }

        System.out.println("Book not found or not checked out.");

    }

}

1. LibrarySystem.java

import java.util.Scanner;

public class LibrarySystem {

    public static void main(String[] args) {

        Library library = new Library();

        Scanner scanner = new Scanner(System.in);

        String choice;

        // Adding some books to the library

        library.addBook(new Book("The Great Gatsby", "F. Scott Fitzgerald"));

        library.addBook(new Book("1984", "George Orwell"));

        library.addBook(new Book("To Kill a Mockingbird", "Harper Lee"));

        do {

            System.out.println("\nLibrary Menu:");

            System.out.println("1. View Available Books");

            System.out.println("2. Check Out a Book");

            System.out.println("3. Return a Book");

            System.out.println("4. Exit");

            System.out.print("Choose an option: ");

            choice = scanner.nextLine();

            switch (choice) {

                case "1":

                    library.viewAvailableBooks();

                    break;

                case "2":

                    System.out.print("Enter the title of the book to check out: ");

                    String checkOutTitle = scanner.nextLine();

                    library.checkOutBook(checkOutTitle);

                    break;

                case "3":

                    System.out.print("Enter the title of the book to return: ");

                    String returnTitle = scanner.nextLine();

                    library.returnBook(returnTitle);

                    break;

                case "4":

                    System.out.println("Exiting...");

                    break;

                default:

                    System.out.println("Invalid choice. Please try again.");

            }

        } while (!choice.equals("4"));

        scanner.close();

    }

}

OUTPUTS:

A screenshot of a computer

Description automatically generatedFig showing different stages in CI/CD pipeline created

A screenshot of a computer

Description automatically generated

Fig showing changes made to the source code using CI/CD pipeline

A computer screen with white text

Description automatically generated

Fig showing execution of source code in command prompt

A screenshot of a computer

Description automatically generated

Fig showing the github repository used for continous integration

# CONCLUSION

The Library Management System project demonstrates how DevOps practices enhance software development by automating key stages like building, testing, and deployment. With features such as book and user management, automated checkouts, and JUnit testing, this system is both practical and reliable.  
  
By using tools like Jenkins, GitHub, and JUnit, we created a streamlined CI/CD pipeline that ensures consistency and efficiency. This project highlights the importance of automation and collaboration in delivering scalable and maintainable software solutions. We are confident that this system can serve as a foundation for further enhancements, such as integrating a user-friendly web interface or extending functionality for multi-library management.

# REFERENCES:

1. Jenkins Documentation: <https://www.jenkins.io/doc/>
2. JUnit User Guide: <https://junit.org/junit5/docs/current/user-guide/>
3. GitHub Documentation: <https://docs.github.com/>
4. Project Implementation Files and Logs.