

INTERNSHIP REPORT
ON
ORDER MANAGEMENT SYSTEM & CI/CD Pipeline

A Report submitted to the Rajiv Gandhi University of Knowledge Technologies in fulfillment of the degree of Bachelor of Technology in Computer Science.

By
Shaik Mahammad Tippu Sulthan
R170444

Under the supervision of
SHABANA SHAIK
Assistant Professor
Computer Science Engineering



Idupulapaya, Vempalli,
Kadapa - 516 330, Andhra Pradesh, India

CERTIFICATE

This is to certify that the report entitled "Long Term Internship on Order Management System And CI/CD pipeline" submitted by Shaik Mahammad Tippu Sulthan (R170444) in fulfillment of the requirements for the award of Bachelor of Technology in Computer Science is a bonafide work carried out by him under my supervision and guidance. The report has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Shabana Shaik,
Project Internal Guide,
CSE,
RGUKT, R.K Valley.

Satyanandaram N,
Head of the Department,
CSE,
RGUKT, R.K Valley.

DECLARATION

I Shaik Mahammad Tippu Sulthan hereby declare that this report entitled “Long Term Internship on Order Management System And CI/CD pipeline” submitted by me under the guidance and supervision of Shabana Shaik is a bonafide work. I also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date: 30-04-2022

Place: RK Valley

Shaik Mahammad Tippu Sulthan
(R170444)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.

I am extremely grateful to our Director.Prof. K. SANDHYA RANI for fostering an excellent academic climate in our institution.

I also express my sincere gratitude to our respected Head of the Department Mr.N.SATYANANDARAM for his encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project.

I would like to convey thanks to our guide at college Ms.SK.SHABANA for her guidance, encouragement, co-operation and kindness during the entire duration of the course and academics.

My sincere thanks to all the members who helped me directly and indirectly in the completion of project work. I express my profound gratitude to all our friends and family members for their encouragement.

TABLE OF CONTENTS

Title	Page
Abstract	
Introduction	
Order Management System	
Flip Kart (OMS) Integration	
CI/CD pipeline	
Python	
Django	
Git & Github	
Conclusion	

ABSTRACT

1.Order Management System(OMS):-

An Order Management System (OMS) is a software application designed to facilitate and streamline the process of managing customer orders from initiation to fulfillment. The system typically provides features such as inventory management, order processing, shipping, and billing. The OMS serves as a central hub for managing all order-related activities, including capturing and processing customer orders, managing inventory levels, allocating resources for fulfillment, and providing real-time visibility into order status.

ABSTRACT

2.CI/CD Pipeline: -

A CI/CD pipeline is a crucial component of modern software development that involves automated testing, continuous integration, and continuous delivery. The project aims to design and implement an effective CI/CD pipeline for a software application. The pipeline will incorporate various tools and technologies such as Git, Jenkins, Docker, and Kubernetes, among others. The project will focus on creating a seamless workflow for developers to commit code changes, run automated tests, build and package the application, and deploy it to the production environment. The implementation of the CI/CD pipeline will improve the quality of the software application by catching bugs early in the development cycle, reducing deployment time, and increasing overall efficiency.

INTRODUCTION

1.Order Management System(OMS):-

An Order Management System (OMS) is a critical software application used by businesses to manage their sales orders, inventory, and fulfillment processes. An OMS helps companies to streamline their sales operations and provides a centralized platform for managing customer orders, shipments, and returns. It enables businesses to efficiently process orders from multiple channels, including online marketplaces, physical stores, and direct sales.

An OMS is designed to provide businesses with real-time visibility into their order and inventory data, allowing them to make data-driven decisions and respond to customer needs promptly. It can help businesses optimize their order processing workflows, reduce order processing times, and improve the accuracy of order fulfillment. With an OMS in place, businesses can provide their customers with a seamless ordering experience, from placing orders to receiving products.

This project aims to develop an OMS that will enable businesses to manage their orders, inventory, and fulfillment processes efficiently. The system will be designed to be scalable and flexible, allowing businesses to adapt to changing market conditions and customer demands. The OMS will incorporate features such as real-time order tracking, inventory management, and reporting capabilities. It will be built using modern technologies, ensuring that it is secure, reliable, and easy to use. The goal is to provide businesses with a solution that will help them streamline their operations and improve their bottom line.

One of the key benefits of an OMS is its ability to automate and optimize the order fulfillment process. By streamlining and automating processes, businesses can reduce errors and delays, improve customer satisfaction, and increase operational efficiency.

Technologies Used

An Order Management System (OMS) is a complex software application that requires the integration of various technologies to function effectively. Below are some of the technologies commonly used in the development of an OMS:

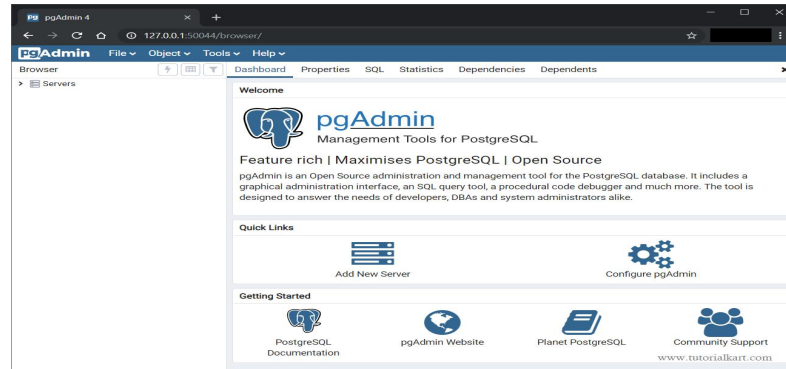
- **PostgreSQL:-**



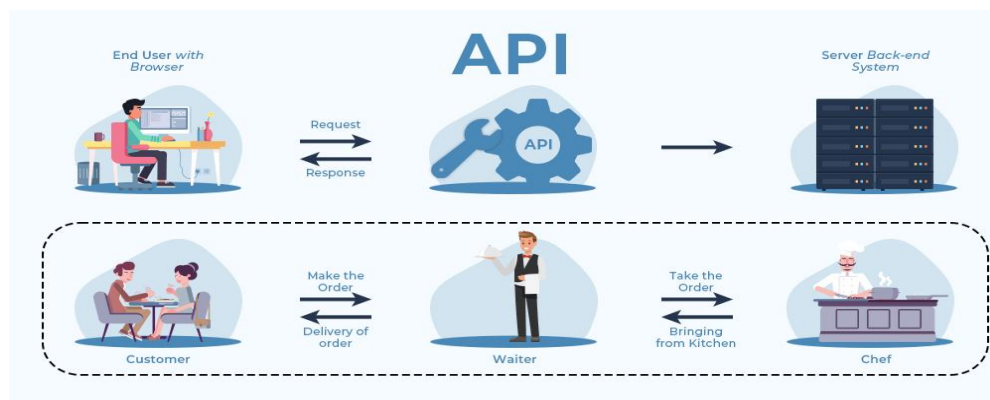
An OMS needs a robust database management system to store and manage data related to orders, inventory, customers, and other aspects of the business. We choose **PostgreSQL** which is a popular open-source relational database management system (RDBMS) that is widely used in modern web applications.

PostgreSQL is known for its robustness, scalability, and adherence to SQL standards. It is often used in applications that require complex data structures, support for advanced features such as triggers and stored procedures, and the ability to handle large datasets. PostgreSQL supports a wide range of programming languages, provides various extension

modules, including support for JSON, XML, and full-text search, which make it a versatile choice for developers working on a wide range of applications.



● Application Programming Interfaces (APIs):-



An OMS needs to integrate with other applications such as **eCommerce platforms**, **payment gateways**, **shipping carriers**, and **accounting software**. APIs enable the OMS to communicate and exchange data with these applications.

- **Cloud Computing Platforms:-**



An OMS can be deployed on cloud platforms such as Amazon Web Services (AWS), Google Cloud, or Microsoft Azure. Cloud computing platforms offer scalable infrastructure, which allows businesses to easily expand or reduce their computing resources depending on their needs. We are using **HETZNER** cloud services.

- **Front-end Technologies:-**

An OMS requires an intuitive user interface that enables users to manage orders, inventory, and other aspects of the business.

Common front-end technologies used in the development of an OMS include HTML, CSS, and JavaScript. We have used Angular JS as Front-end.

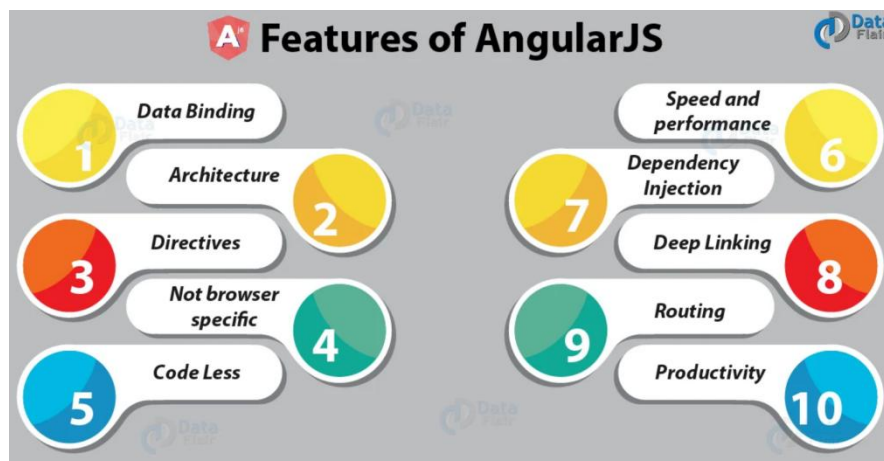
- **Angular JS:-**



AngularJS is an open-source, front-end JavaScript framework that was developed by Google. It was released in 2010 and has since gained immense popularity among web developers due to its robustness and versatility. AngularJS uses a Model-View-Controller (MVC) architecture that makes it easy to build dynamic web applications. It simplifies the

development process by providing a set of tools and directives that allow developers to build responsive and interactive user interfaces.

AngularJS uses two-way data binding, which allows changes made to the model to be automatically reflected in the view and vice versa. It also provides a rich set of directives that can be used to extend HTML with custom attributes and elements. Dependency injection is another key feature that makes it easy to manage and test components. AngularJS is designed to work seamlessly with other web technologies and libraries, such as Node.js, jQuery, and Bootstrap. It is also highly extensible, with a large and active community of developers constantly creating new plugins, modules, and extensions. Overall, AngularJS is a powerful and versatile framework that provides developers with a range of tools and features for building robust and scalable web applications.



● Back-end Technologies:-

The back-end is the server-side of the OMS that processes requests, stores data, and interacts with other systems. The back-end typically

includes database management, order processing, inventory management, and integration with third-party systems such as payment gateways and shipping providers. Here we have used **Django**.

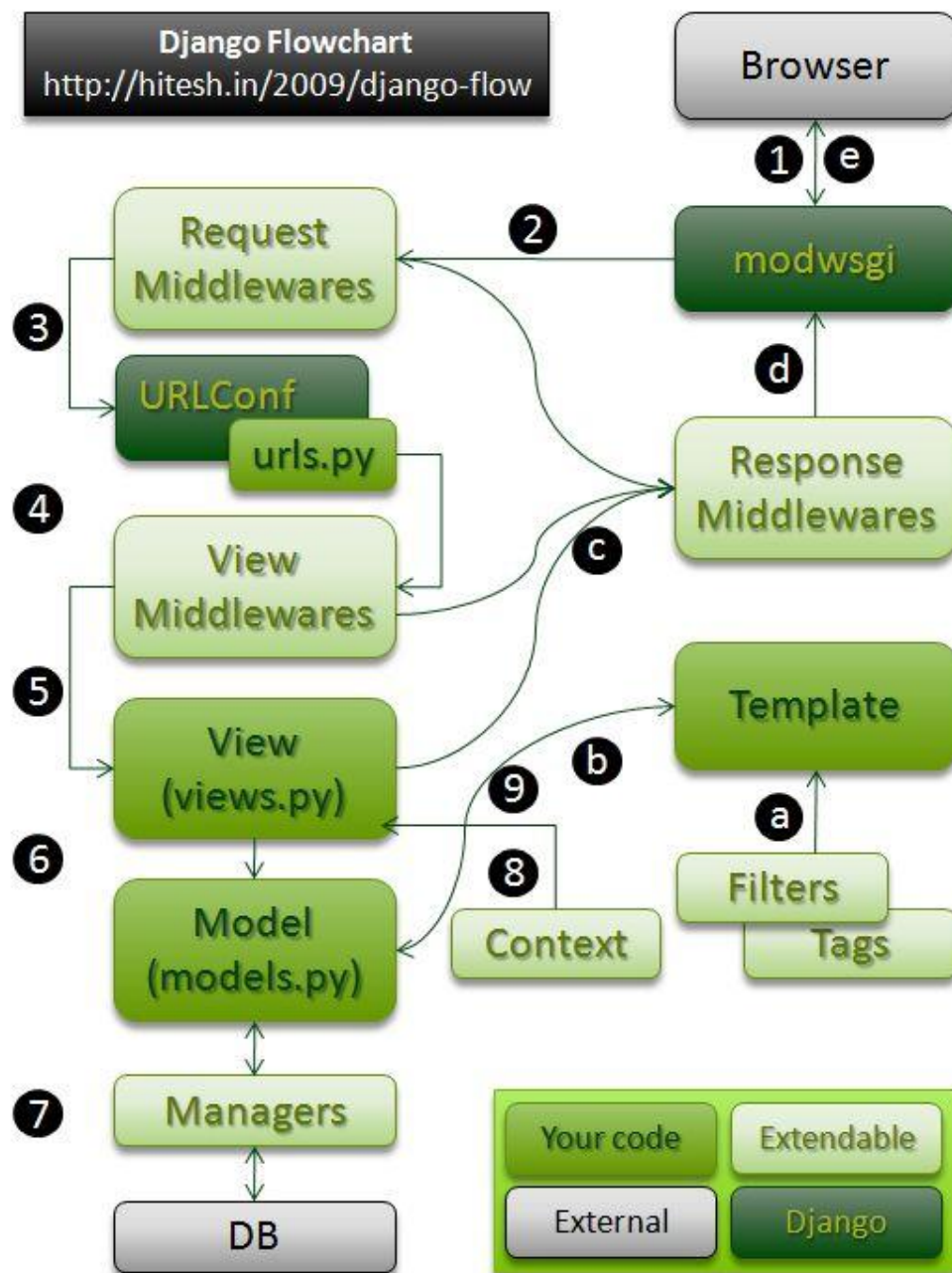
- **Django:-**



Django is a high-level Python web framework that enables developers to build powerful and scalable web applications quickly and efficiently. It follows the model-view-controller (MVC) architectural pattern, which separates the application into three components: the model (data layer), the view (user interface layer), and the controller (logic layer).

Django provides a rich set of features, including an object-relational mapper (ORM), an automatic admin interface, a powerful URL routing system, and built-in security features like CSRF protection and user authentication. It also offers support for multiple databases, caching, and internationalization. Django's main goal is to help developers write clean, reusable, and maintainable code. It emphasizes the use of best practices, such as the Don't Repeat Yourself (DRY) principle and the Convention over Configuration approach. Django also encourages the use of third-party packages and plugins to extend its functionality and make development even faster. Overall, Django is a robust and reliable web framework that can handle complex web applications with ease. Its focus on code quality and best practices makes it a popular choice among developers who value efficiency and scalability.

Django web applications typically group the code that handles each of these steps into separate files:



2.CI/CD Pipeline:-

A CI/CD pipeline is a set of automated processes that allows developers to continuously build, test, and deploy code changes to a software application. The acronym "CI/CD" stands for Continuous Integration and Continuous Deployment/Delivery.

A CI/CD pipeline is a sequence of automated steps that includes continuous integration, testing, and deployment of code changes. This pipeline helps to ensure that the software is delivered reliably and quickly with a minimum of errors and downtime.

The CI/CD pipeline begins with code changes in a source code repository. When code changes are pushed to the repository, the CI/CD pipeline automatically builds and tests the changes. If the build and tests pass successfully, the pipeline will deploy the changes to a staging environment where they can be further tested before being released to production.

Continuous Integration is the first step in the CI/CD pipeline. It involves developers regularly committing their code changes to a shared repository. The CI server then automatically builds and tests the changes. This ensures that code is integrated frequently and conflicts between different changes are caught early.

Continuous Delivery is the next step in the pipeline. It automates the process of releasing code changes to production-like environments for testing, and can include manual approval steps for releasing to production.

Continuous Deployment is the final step in the pipeline, where code changes are automatically released to production as soon as they pass all the automated tests.

CI/CD pipelines can be implemented using a variety of tools and technologies, such as Jenkins, GitLab CI/CD, CircleCI, Travis CI, or GitHub Actions. The pipeline can be customized to suit the needs of the software development team, and can include additional steps such as security testing, performance testing, and code analysis.

Technologies Used

There are several technologies and tools that can be used for building a CI/CD (Continuous Integration/Continuous Deployment) pipeline. But we are using the following:

- **Jenkins:**

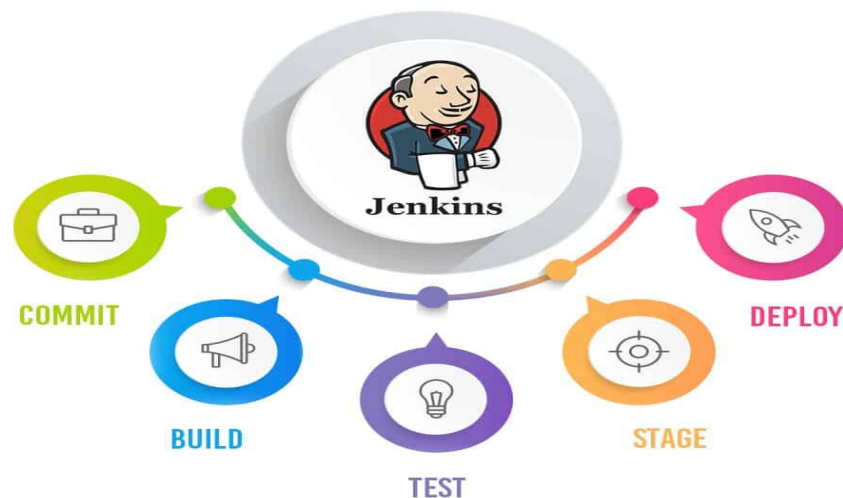


Jenkins is a popular open-source automation server that is used for building, testing, and deploying software. It is written in Java and supports a vast number of plugins and integrations with other tools, making it very flexible.

Jenkins is used for implementing Continuous Integration (CI) and Continuous Deployment (CD) pipelines. The primary goal of CI is to ensure that software changes can be integrated and tested frequently and reliably, while CD automates the deployment of software changes to

production. By using Jenkins, developers can automate repetitive tasks, reduce errors, and increase the speed and efficiency of the development process.

Jenkins is easy to set up and can be installed on any platform that supports Java. It can be configured to work with a wide range of technologies and tools, including Git, Maven, Gradle, Docker, and many others. Jenkins can also be integrated with various testing frameworks, such as JUnit and Selenium. Jenkins provides a web-based interface for configuring jobs and pipelines. Jobs represent individual steps in a pipeline, such as building, testing, and deploying software. Pipelines are sequences of jobs that define the stages of a software delivery process. Jenkins also provides various features for monitoring and reporting on the status of pipelines, such as build history, test results, and code coverage metrics.



- **Webhooks:**

Webhooks are a way for web applications to communicate with each other in a real-time, event-driven manner. A webhook is essentially a

POST request sent by one application to another application when a particular event occurs. The receiving application can then take action based on the data contained in the webhook payload. Webhooks are typically implemented using HTTP requests, although other protocols such as FTP and email can also be used. The payload of a webhook can be in various formats, such as JSON or XML, and can contain any data that the sending application chooses to include.

One advantage of using webhooks is that they can reduce the need for polling, where an application periodically checks for updates from another application. With webhooks, the receiving application is notified immediately when an event occurs, which can lead to faster and more efficient processing.

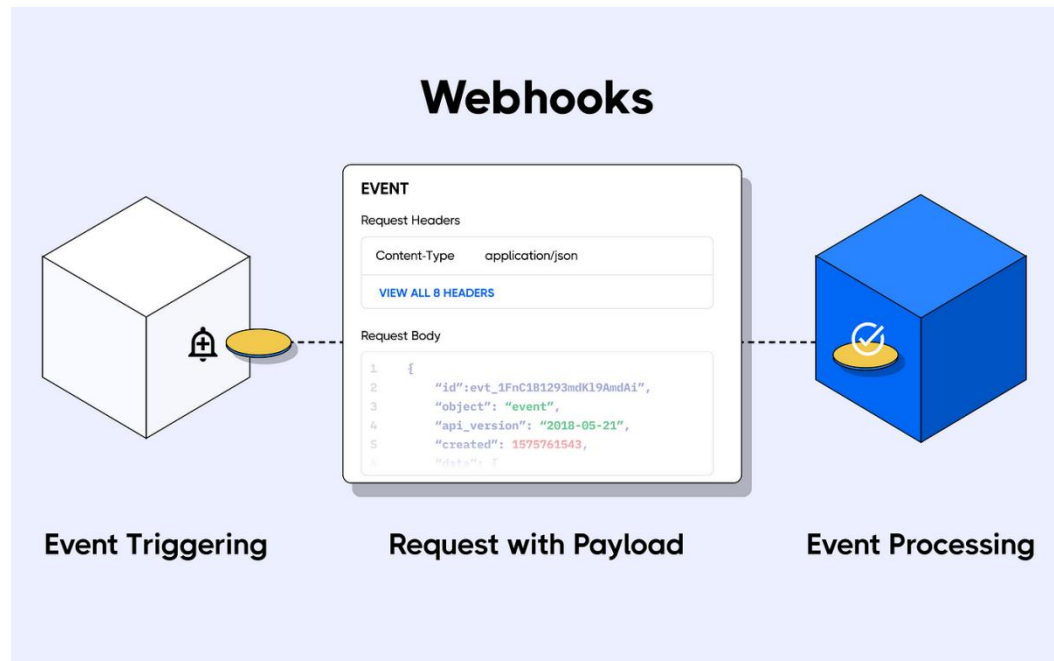
Webhooks in Jenkins are a mechanism to trigger builds automatically when changes are made in an external repository or service. Webhooks are a way for external systems to notify Jenkins of an event, such as a code push or pull request, so that Jenkins can start a build or other automated task.

To use webhooks in Jenkins, you first need to configure the webhook URL in the external system, such as GitHub or Bitbucket. The webhook URL should point to the Jenkins job that you want to trigger.

Next, you need to configure the Jenkins job to listen for incoming webhooks. This can be done using the "Generic Webhook Trigger" plugin, which can be installed through the Jenkins plugin manager. Once installed, you can configure the plugin to parse the payload of the incoming webhook and extract any relevant information, such as the branch name or commit hash. Finally, you can use the information from the webhook payload to start a build or other automated task in Jenkins. This can be done using standard Jenkins build steps, such as executing a

shell command or running a Maven build.

Overall, webhooks in Jenkins provide a powerful way to automate your development workflow and ensure that builds are triggered automatically when changes are made in your source code repository or other external systems.



- **Git Hub:**

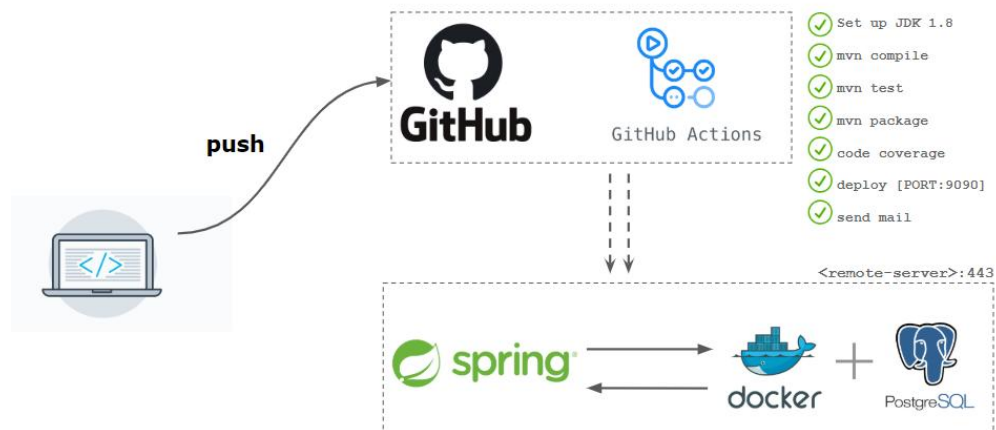
GitHub plays a crucial role in the Continuous Integration/Continuous Deployment (CI/CD) pipeline by providing a platform for hosting and managing source code, tracking changes, and enabling collaboration among team members.

GitHub provides various tools and services that help developers to implement CI/CD pipelines, such as GitHub Actions, GitHub Packages,

and GitHub Pages. GitHub Actions is a powerful feature that enables developers to automate their workflows and run tests and build processes for their code directly from their GitHub repositories.

With GitHub Actions, developers can set up and configure continuous integration and continuous deployment pipelines that automatically build, test, and deploy their code. GitHub Packages is a package hosting service that allows developers to host and manage their packages and dependencies directly from their GitHub repositories. Developers can use GitHub Packages to store and share their packages with other team members or the wider community. GitHub Pages is a free web hosting service provided by GitHub that allows developers to create and host static websites directly from their GitHub repositories. With GitHub Pages, developers can deploy their websites automatically as part of their CI/CD pipeline.

Overall, GitHub plays a critical role in enabling developers to implement a comprehensive and efficient CI/CD pipeline, which is essential for modern software development.



- **Docker:**



Docker is a platform that enables developers to build, ship, and run applications in containers. Containers are a lightweight and portable way to package software and its dependencies into a single, self-contained unit that can run consistently across different environments.

With Docker, developers can create containers that include all the necessary components of their application, such as the operating system, libraries, and dependencies. These containers can then be shipped to different environments, such as testing, staging, and production, without worrying about compatibility issues.

Docker is commonly used in Continuous Integration/Continuous Deployment (CI/CD) pipelines to create consistent and reproducible environments for building and deploying applications. Here are some of the ways Docker is used in a typical CI/CD pipeline:

Build: Docker is used to build the application and its dependencies into a container image. This image can then be used to create identical containers in any environment.

Test: Docker is used to create isolated testing environments where tests can be run against the application. Each test run can be performed in a fresh container, ensuring that the results are consistent and not influenced by previous runs.

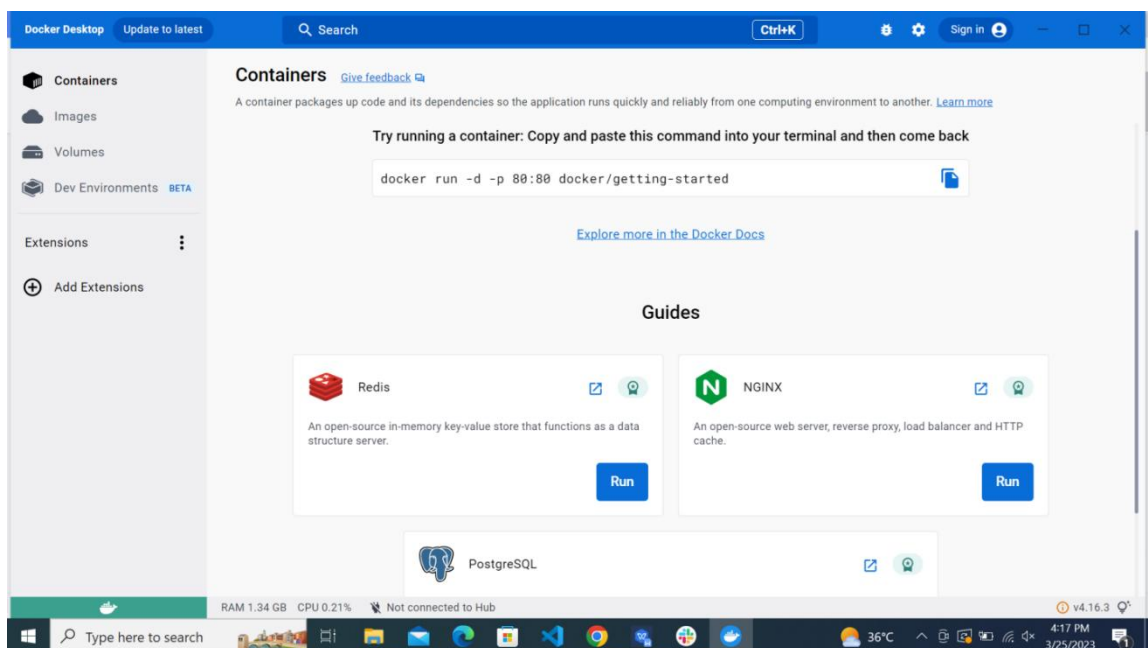
Deploy: Docker is used to create a deployment package that includes the application and its dependencies in a container image. This image can then be deployed to any environment that supports Docker,

making the deployment process more consistent and repeatable.

Versioning: Docker images can be versioned, making it easy to track changes to the application over time. This allows developers to roll back to a previous version of the application if necessary.

Infrastructure as Code: Docker images can be treated as infrastructure as code, which means that the configuration for the container environment can be versioned and tracked along with the application code.

Overall, Docker plays a crucial role in a modern CI/CD pipeline by providing consistent and reproducible environments for building, testing, and deploying applications. By using Docker, developers can reduce the risk of errors and inconsistencies in the deployment process, and make it easier to manage the entire software development lifecycle.



Docker Desktop

Order Management System(OMS):

An order management system is a software platform used by businesses to manage their sales and fulfillment processes. It helps to streamline the entire order process from initial order placement through to delivery and returns. An effective order management system can improve order accuracy, reduce processing time, and enhance customer satisfaction. It provides businesses with real-time visibility into inventory levels, order status, and customer data, allowing them to make informed decisions and optimize their operations. Overall, an order management system is a vital tool for businesses seeking to improve their order processing and fulfillment capabilities.

Why OMS is Needed?

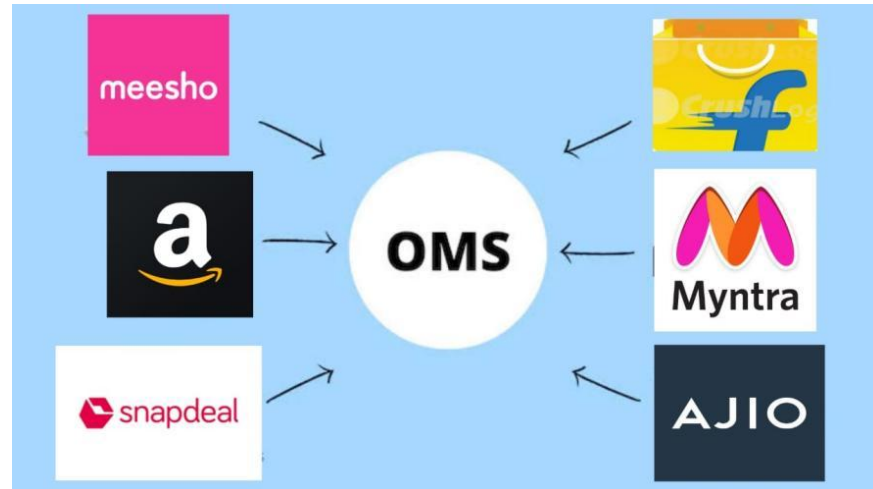
A B2B seller can sell their products on multiple e-commerce platforms, including Flipkart, Amazon, Myntra, Ajio, and others. However, managing orders across these platforms can be challenging. For instance, the vendor may need to log in to each platform separately to update their stock, obtain order details, and manage inventory. This can be a difficult task for an individual to ensure that all the information is verified accurately.

Let us consider an example:

A vendor has 10 shirts to sell and wants to list them on multiple e-commerce platforms. To do this, they must manually update the stock as 10 on each platform. If the vendor receives 5 orders on Flipkart, they must update the inventory to reflect 5 on every other platform. If they forget to update the stock on Myntra, the inventory will still show as 10 and they may receive orders for more shirts than they actually have

available. Failing to fulfill customer orders can result in penalties and even lead to being blocked from selling on certain platforms. This is where an Order Management System comes in. It integrates all the B2B marketplaces into one application, making it easier for the vendor to manage their inventory and fulfill customer orders.

We can integrate all the e-commerce applications into one and perform all the supply chain operations through that application and that integrated application is called OMS



An eCommerce order management system lets a business consolidate and manage orders coming from multiple sales channels in a single interface. An order management system is a comprehensive solution that automates data transfer between departments, aggregates order data, and lets you manage and adjust orders easily. These features are important for all types of eCommerce businesses as orders can come in through a variety of channels at any time.



Flip Kart (OMS) Integration:

Integrating Flipkart's OMS with our WMS involves several steps, including registration, API integration, testing, and production launch. Here are the high-level steps to integrate Flipkart's OMS:

1. Register on Flipkart Marketplace: Before integrating the OMS, you need to register as a seller on Flipkart Marketplace. Once you have registered, you can access the seller dashboard and API documentation.
2. Access API documentation: Flipkart provides API documentation to help sellers understand how to integrate with their OMS. The documentation includes details on API endpoints, data formats, authentication, and other technical details.

3. Integrate with the API: Once you have understood the API documentation, you can start integrating your inventory management system with Flipkart's OMS API. You will need to use REST APIs to communicate with Flipkart's platform and exchange order and inventory data.

4. Test the integration: After integrating your system with the Flipkart OMS API, you need to test your integration to ensure that it is working correctly. Flipkart provides a sandbox environment for testing your integration, where you can create test orders and verify the data exchange between your system and Flipkart's platform.

5. Launch in production: Once you have tested your integration, you can launch it in production. Before going live, make sure to verify that your system is ready to handle production traffic and that you have tested all the critical scenarios.

Overall, integrating Flipkart's OMS requires technical expertise and careful testing to ensure that your integration works correctly. Flipkart provides extensive documentation and support to help sellers with the integration process.

Links:

- <https://seller.flipkart.com/>
- <https://seller.flipkart.com/api-docs/index.html>

CI/CD Pipeline:

A CI/CD (Continuous Integration/Continuous Deployment) pipeline is a process that automates the building, testing, and deployment of software applications. The pipeline allows developers to quickly and efficiently deliver high-quality software to their users. In the CI phase, code changes are integrated into a shared repository multiple times per day. Automated tests are run to check the changes, and if any issues are found, they are reported back to the developer for correction. This ensures that any errors are caught early in the development process. In the CD phase, the code is automatically deployed to various environments, such as staging or production, after it passes all the automated tests. This automation helps to reduce the time and effort needed for deployment and minimize the chance of human error. Overall, a CI/CD pipeline helps to streamline the software development process, reduce errors, and ensure the software is delivered quickly and efficiently.

CI/CD (Continuous Integration/Continuous Delivery) pipeline is a set of automated processes that help software development teams to continuously integrate, test, and deploy their code changes into production. It is needed for several reasons:

1. Speed up software development: With CI/CD pipeline, developers can release code changes more frequently and quickly, which allows them to keep up with the fast pace of modern software development.
2. Early detection of bugs: CI/CD pipeline automates the testing process, which helps in detecting bugs and issues early on in the development process, before they become bigger problems.
3. Consistency and reliability: CI/CD pipeline provides a consistent and

reliable way to deploy software changes, ensuring that the software works as expected across different environments.

4. Collaboration: CI/CD pipeline encourages collaboration between developers, testers, and other stakeholders, enabling them to work together seamlessly towards a common goal.

5. Increased efficiency: By automating the build, test, and deployment process, CI/CD pipeline reduces the amount of time and effort required to release software, which leads to increased efficiency.

CI/CD pipeline is popular because it brings many benefits to the software development process, including increased speed, improved quality, and enhanced collaboration. It is now considered a best practice in modern software development, and is widely adopted by development teams around the world.

Here are the general steps to develop CI/CD using Jenkins webhook:

1. Install and configure Jenkins on your server.
2. Create a Jenkins job for your application.
3. Install the necessary plugins in Jenkins to support webhook integration.
4. Configure Jenkins to allow webhook triggers. This usually involves setting up a webhook URL and configuring the payload format.
5. Configure your source code repository to send webhooks to Jenkins when changes are made.
6. Test the webhook integration to ensure that Jenkins can receive and process webhook requests.

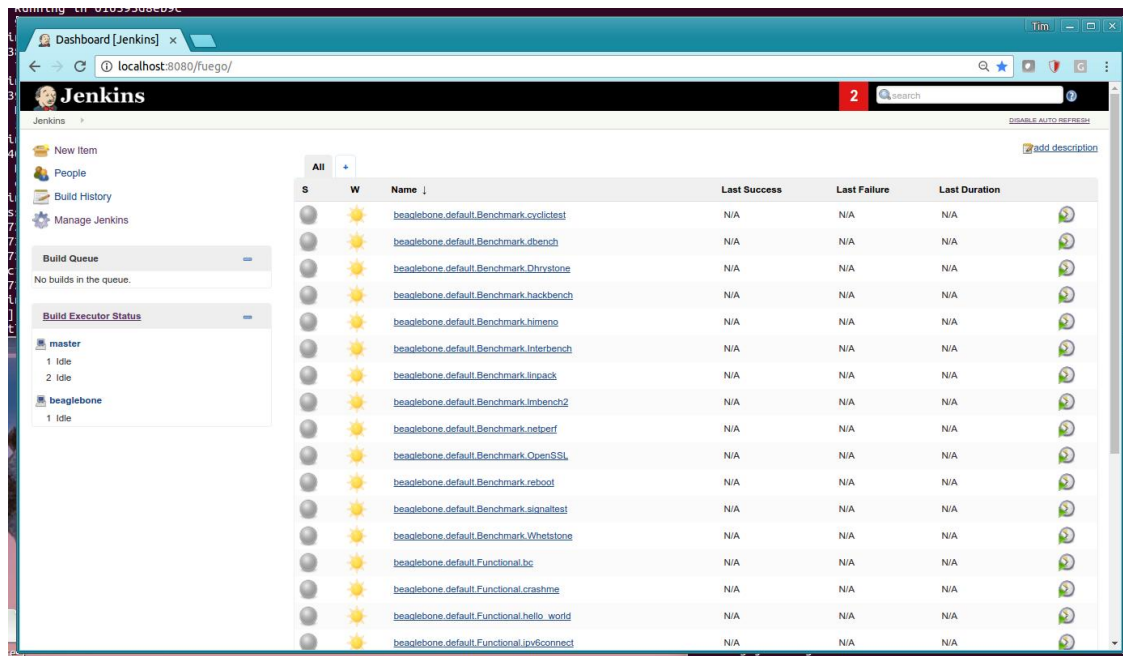
7. Configure your Jenkins job to build, test, and deploy your application automatically when a webhook trigger is received.

8. Test the automated build and deployment process to ensure that it works as expected.

9. Set up notifications to receive alerts when builds or deployments fail. Monitor your Jenkins job to ensure that it continues to function properly and to make any necessary changes or updates.

Overall, the key to success with CI/CD using Jenkins webhook is to thoroughly test and refine your setup over time, and to continuously monitor and improve your process to optimize for speed, reliability, and accuracy.

Jenkins:

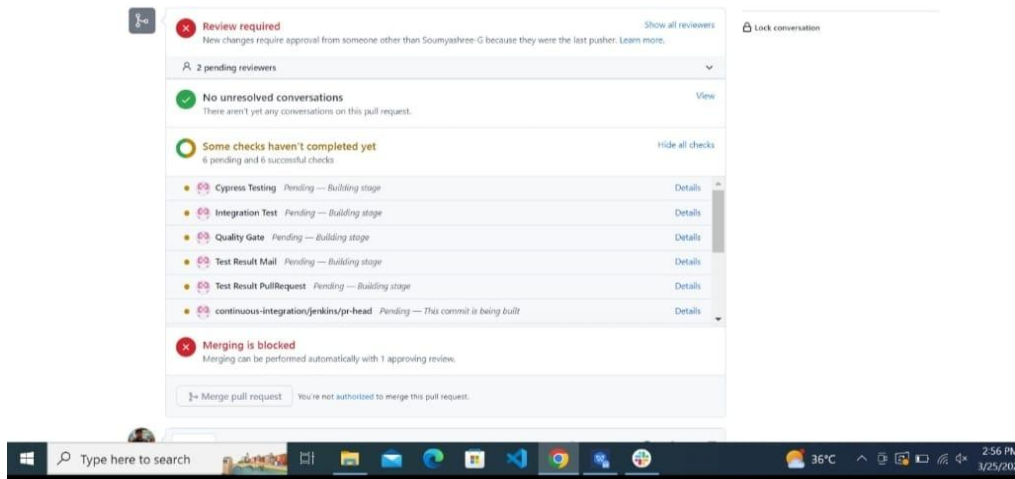


The screenshot shows the Jenkins Dashboard interface. On the left, there is a sidebar with navigation links: 'New Item', 'People', 'Build History', and 'Manage Jenkins'. Below these, there are sections for 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing 'master' with 1 idle executor and 'beaglebone' with 1 idle executor). The main area displays a table of jobs. The table has columns for 'S' (Status), 'W' (Webhook), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. The jobs listed are all benchmarks and functional tests, all with a status of 'N/A' for the last success and failure, and a duration of 'N/A'. Each job has a green checkmark icon in the 'S' column.

S	W	Name	Last Success	Last Failure	Last Duration
✓	✓	beaglebone.default.Benchmark.cycloctest	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.dbench	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.Dhrystone	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.hackbench	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.himeno	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.interbench	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.linkpack	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.lmbench2	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.netperf	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.OpenSSL	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.reboot	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.signalsstest	N/A	N/A	N/A
✓	✓	beaglebone.default.Benchmark.Whetstone	N/A	N/A	N/A
✓	✓	beaglebone.default.Functional.btc	N/A	N/A	N/A
✓	✓	beaglebone.default.Functional.crashme	N/A	N/A	N/A
✓	✓	beaglebone.default.Functional.hello_world	N/A	N/A	N/A
✓	✓	beaglebone.default.Functional.ipv6connect	N/A	N/A	N/A

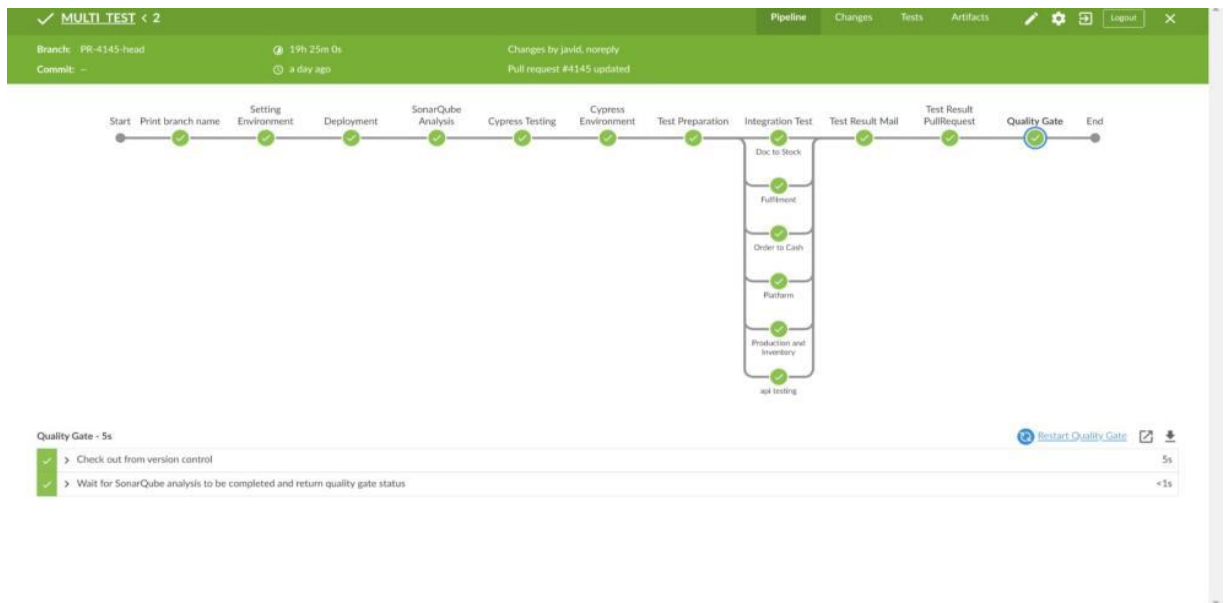
Jenkins View

In Git Hub when PULL REQUEST raised:



Building Stage of CI/CD

CI/CD pipeline:



CI/CD Pipeline view

Testing Report(Integration Testing):

Total	911	14	117	348	29	164	93	146
Passed	893	14	115	347	29	164	88	136
Failed	18	0	2	1	0	0	5	10
Skipped	0	0	0	0	0	0	0	0
Pending	0	0	0	0	0	0	0	0

Module Wise Testing Report

After All Checks Passed:

The image shows two screenshots of a GitHub Pull Request interface. Both screenshots display a 'Review required' status at the top, indicating that new changes need approval from someone other than the last pusher, 'Sudhanwa23'. Below this, there is a section for '1 approval' and a 'No unresolved conversations' message. The main section, 'All checks have passed', lists 12 successful checks. The first screenshot shows checks for 'Cypress Environment', 'Cypress Testing', 'Deployment', 'Integration Test', 'Print branch name', and 'Quality Gate'. The second screenshot shows checks for 'Setting Environment', 'SonarQube Analysis', 'Test Preparation', 'Test Result Mail', 'Test Result PullRequest', and 'continuous-integration/jenkins/pr-head'. Each check is marked with a green checkmark and a 'Details' link.

Review required [Show all reviewers](#)
New changes require approval from someone other than Sudhanwa23 because they were the last pusher. [Learn more.](#)

✓ 1 approval [View](#)

✓ **No unresolved conversations**
There aren't yet any conversations on this pull request. [View](#)

✓ **All checks have passed** [Hide all checks](#)
12 successful checks

- ✓ **Cypress Environment** — Stage built successfully [Details](#)
- ✓ **Cypress Testing** — Stage built successfully [Details](#)
- ✓ **Deployment** — Stage built successfully [Details](#)
- ✓ **Integration Test** — Stage built successfully [Details](#)
- ✓ **Print branch name** — Stage built successfully [Details](#)
- ✓ **Quality Gate** — Stage built successfully [Details](#)

Review required [Show all reviewers](#)
New changes require approval from someone other than Sudhanwa23 because they were the last pusher. [Learn more.](#)

✓ 1 approval [View](#)

✓ **No unresolved conversations**
There aren't yet any conversations on this pull request. [View](#)

✓ **All checks have passed** [Hide all checks](#)
12 successful checks

- ✓ **Setting Environment** — Stage built successfully [Details](#)
- ✓ **SonarQube Analysis** — Stage built successfully [Details](#)
- ✓ **Test Preparation** — Stage built successfully [Details](#)
- ✓ **Test Result Mail** — Stage built successfully [Details](#)
- ✓ **Test Result PullRequest** — Stage built successfully [Details](#)
- ✓ **continuous-integration/jenkins/pr-head** — This commit looks good [Details](#)

All Checks Passed and Branch is Ready To Merge

● **Python:**

Python is a high-level, interpreted, interactive and object-oriented scripting language Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python has a large and active community of developers who contribute to its growth and development, and there are numerous libraries and frameworks available that make it easy to build complex applications. One of the key features of Python is its readability, which is achieved through the use of whitespace to delimit code blocks, and a syntax that emphasizes simplicity and clarity. This makes it easier to write, read, and maintain code, even for those with little experience in programming.

Reference:

<https://docs.python.org/3/>

● **Django:**

Django is a high-level web framework for building web applications quickly and efficiently in Python. It was created with the goal of making web development easier, more productive, and more secure. One of the main benefits of using Django is its built-in features and functionality. It includes a powerful Object-Relational Mapping (ORM) system that makes it easy to work with databases, a templating engine for rendering HTML pages, and an easy-to-use admin interface for managing content. Django follows the Model-View-Controller (MVC) design pattern, but it calls it Model-View-Template(MVT). This architecture helps to separate the data layer from the presentation layer, making it easier to maintain and scale the application. Django also has a large and active community, which means that there are a lot of third-party packages available that can extend its

functionality even further. Overall, Django is a powerful and flexible framework that can be used to build all kinds of web applications, from simple blogs to complex e-commerce sites. Its ease of use and robustness make it a popular choice for developers of all levels of expertise.

Reference:

<https://docs.djangoproject.com/en/4.1/>

● **Git & Github:**

Git is a distributed version control system that allows developers to track changes to their codebase, collaborate with other developers, and manage different versions of their code.

It was created by Linus Torvalds in 2005 to manage the development of the Linux kernel. GitHub is a web-based hosting service that provides a platform for developers to store their Git repositories and collaborate with other developers. It was founded in 2008 and has since become one of the most popular code hosting platforms in the world. Using Git and GitHub, developers can work on their codebase locally and then push changes to a remote repository hosted on GitHub. This allows multiple developers to work on the same codebase and collaborate seamlessly. GitHub also provides features like issue tracking, project management, and pull requests, which makes it easier for developers to collaborate on projects and manage their codebase. Overall, Git and GitHub have become an essential part of modern software development and are widely used by developers all over the world.

Git commands:

- git init
- git clone
- git add
- git status
- git checkout
- git new
- git commit
- git pull
- git push

Reference:

<https://docs.github.com/en>

Conclusion:

To conclude, my long-term internship experience in developing an order management system and implementing a CI/CD pipeline was a challenging and rewarding experience. Through this project, I gained hands-on experience in various aspects of software development, including requirements gathering, design, coding, testing, and deployment. The development of the order management system enabled me to understand the complexities involved in managing orders, tracking inventory, and ensuring timely delivery. Additionally, implementing a CI/CD pipeline helped to automate the software development process, enabling faster releases with improved quality and fewer errors.

Throughout the project, I learned the importance of agile development methodologies, which emphasize collaboration, flexibility, and iterative development. The experience taught me to work in a team environment, communicate effectively with stakeholders, and manage my time efficiently to meet project deadlines.

The project also taught me the importance of collaboration and communication in a team setting. Working with other developers and stakeholders allowed me to better understand how different perspectives and ideas can contribute to the success of a project.

Overall, this internship was a valuable experience that provided me with practical skills and knowledge that will be useful in my future career as a software developer. I am grateful for the opportunity to work on this project and look forward to applying the skills and experience gained to future projects.