

# Godam

(Goods Availability Management)

---

Authored by:

Prashant Sangtani (MT2018082)

Rajeev Pankaj Shukla (MT2018091)

Shubham Darokar (MT20180113)



---

# Contents

Abstract .....	3
Introduction .....	4
Why DevOps? .....	4
About the application .....	7
Software Development Life Cycle .....	9
Agile Methodology.....	11
Project Scope .....	12
Project Architecture .....	13
Source Control Management .....	14
Git .....	14
Build Tools .....	16
Gradle .....	17
Test Tools.....	18
JUnit.....	18
Continuous Integration and Deployment Tools .....	20
Jenkins .....	20
Monitoring Tools .....	22
ELK Stack .....	23
Results .....	25
Future Work .....	26
Conclusion .....	26
List of Figures.....	27
References.....	28

---

# Abstract

Smartphones are mobile devices that travel with their owners and with costs of smartphones taking a dive and power inside them increasing, smartphones have become a great tool to provide increasingly powerful services. Taking advantage of portability and ease of use, we are introducing “Godam” an android application software which helps for the businesses to operate warehouses, where owner can keep the records of available stock.

---

# Introduction

## Why DevOps?

DevOps is a software development approach that focuses on the collaboration between developers and operations, where developers are empowered to own their code from cradle to grave and operations develops tools for automation to be used by their developers. Using a DevOps model, we will share a common goal to quickly deliver quality products and services through more frequent deployment and collaboration.



*Figure 1 : DevOps Culture*

---

DevOps culture believes in the concept of automation to optimize productivity and minimize human errors.

It provides solutions to two main perspectives in the following way:

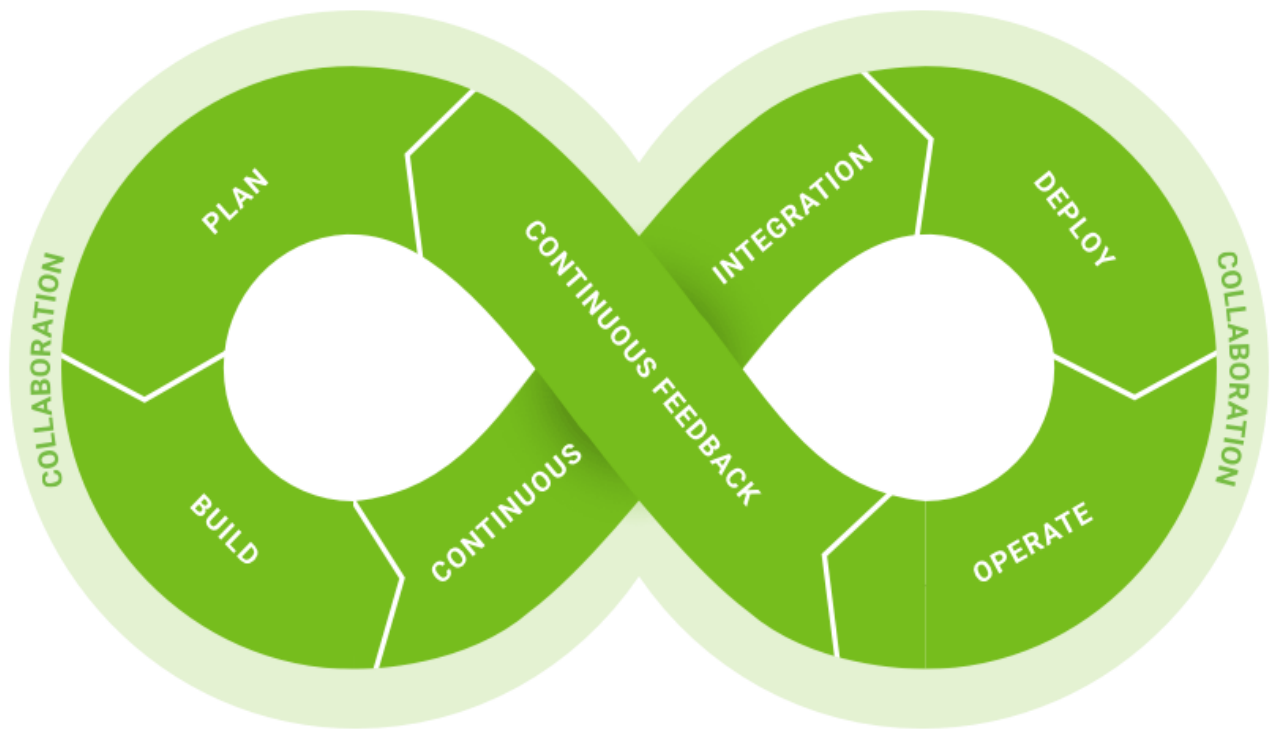
### 1. FROM THE TEAM PERSPECTIVE

- The build and deployment related tasks are always painful in a non-DevOps environment.
- These are the stages where specific environment related issues surge and where all the development that has been implemented by many team members become merged to produce the final version that is going to be delivered.
- These kinds of issues are common in a non-DevOps environment since the DEV and IT teams work separately, with everyone just taking care of his/her own individual responsibilities. With this sort of team organization, people can spend hours and even days trying to reproduce and resolve every emerging problem, something that could end up with frustrated team members, system instability, and delays on deliveries.
- The DevOps culture overcomes these issues by assigning the delivery responsibilities to the entire team and by promoting collaboration and automation throughout the delivery process to minimize pre and post deployment headaches.

DevOps relies on two main automated processes:

- continuous integration
- continuous delivery

both of these are implemented to get faster development cycles without compromising the product's quality. Due to their trusted automated process, DevOps also cares about deployment in diverse environments. In this way, the development team gains more time to concentrate its efforts on new customer necessities rather than on researching deployment related issues. Additionally, due to the shorter development cycles, problem solving tends to be simpler and detected in earlier stages.



*Figure 2 : DevOps Cycle*

## 2. FROM THE CUSTOMERS PERSPECTIVE

- Customers' business can experience negative impacts due to production issues not detected during the development process. The resulting implications can be a system malfunction, wrong accounting calculations, or even system inaccessibility.
- All those aspects could result in negative side effects like user dissatisfaction and a decrease in company revenues that could lead to the loss of millions of dollars.
- Innovation is also a key factor that impacts customers' business effectiveness. The more quickly new features and bugs fixes are released, the faster the adaptation is to changing markets. Delays in this aspect can result in the loss of an opportunity window for a business.
- DevOps improves the customer's business value by providing a continuous delivery of products that satisfy their needs. It also promotes faster delivery of features due to the automation process implemented all over the supply chain.

- 
- Furthermore, system recovery time is minimized due to the shorter delivery cycles and the improved communication and collaboration between developers and operation team members, which is combined with the automated delivery process that verifies that each change is functional and safe to release.
  - As a consequence, time and money can be destined to system innovation rather than to fixing bugs or maintenance related tasks, which allows businesses to grow and compete more efficiently in the market.

## About the application

“Godam” is the acronym for Goods Availability Management and is derived from Hindi word “गोदाम” which means Warehouse. “Godam” can be used in small stores for keeping track of stock easily and save the time.

### INTRODUCTION:

- Godam is software which is helpful for the businesses to operate warehouses, where owner keeps the records of available stocks.
- This project eliminates the paperwork, human faults, manual delay and speed up the process.
- Godam will have the ability to track available stock.
- This is simple, fast and intelligent stock management that can be used by anyone who has a smartphone.

### EXISTING SYSTEM:

- Manual calculation of stock present in any store is time-consuming and very risky.
- The workers cannot maintain the store when there is no owner.

### PROPOSED SYSTEM:

The proposed system is an Android application, it is better than existing system because of following reasons:

- Easily accessible by anyone.
- Overall stock of products can be viewed by owner.
- It doesn't require manpower, so it is not expensive.

---

### ADVANTAGES:

- This Godam project will be a great help for the stores because it is a great difficult task that to manage or get stock information in warehouse and stores. It can also manage the stock by doing operations (add, remove) so that the person will be notified when to get the new stock into the store.

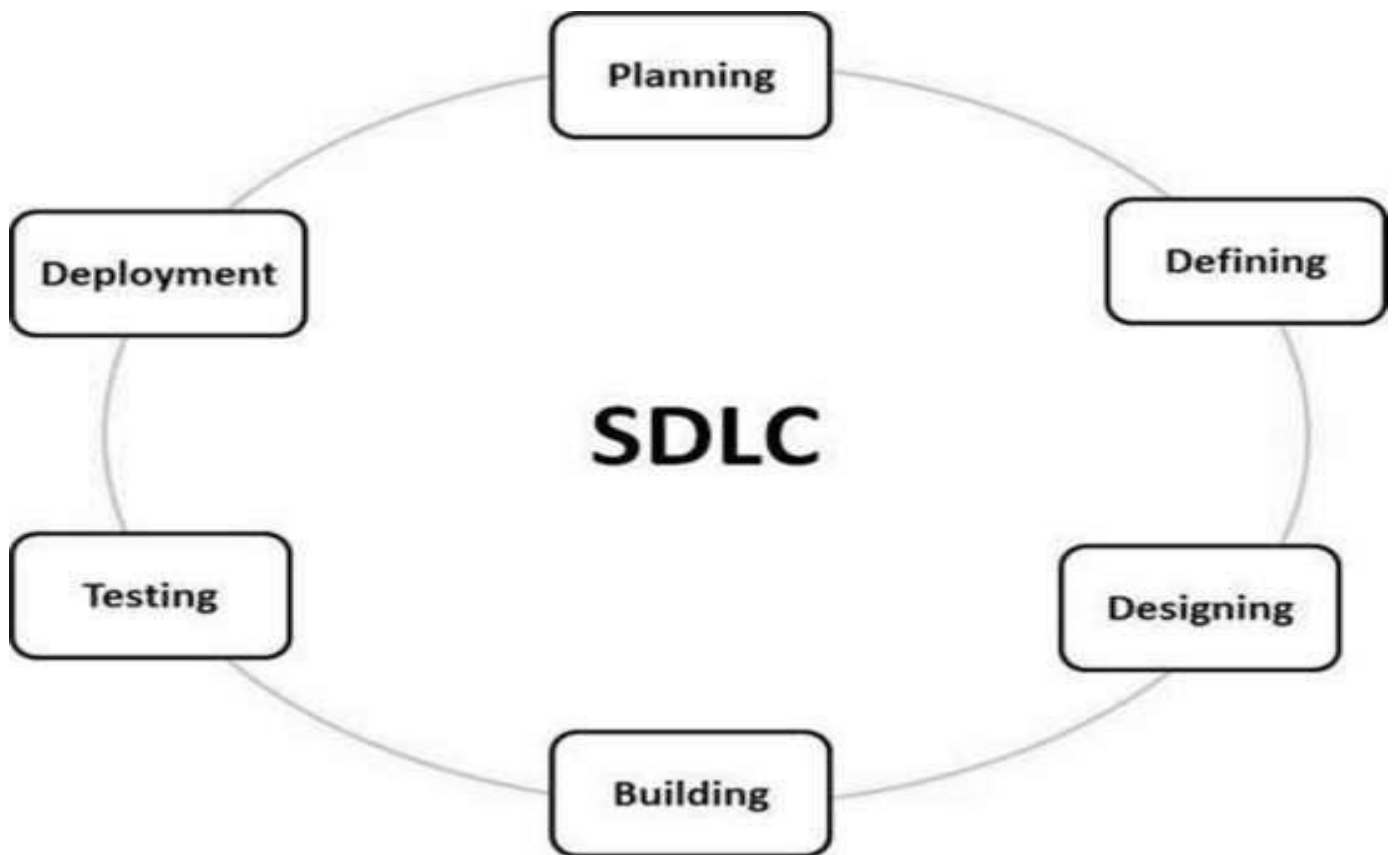


---

# Software Development Life Cycle

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



*Figure 3 : Software Development Life Cycle*

A typical Software Development Life Cycle consists of the following stages –

## Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

---

### Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

### Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS- Design Document Specification.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third-party modules (if any).

### Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

### Stage 5: Testing the Product

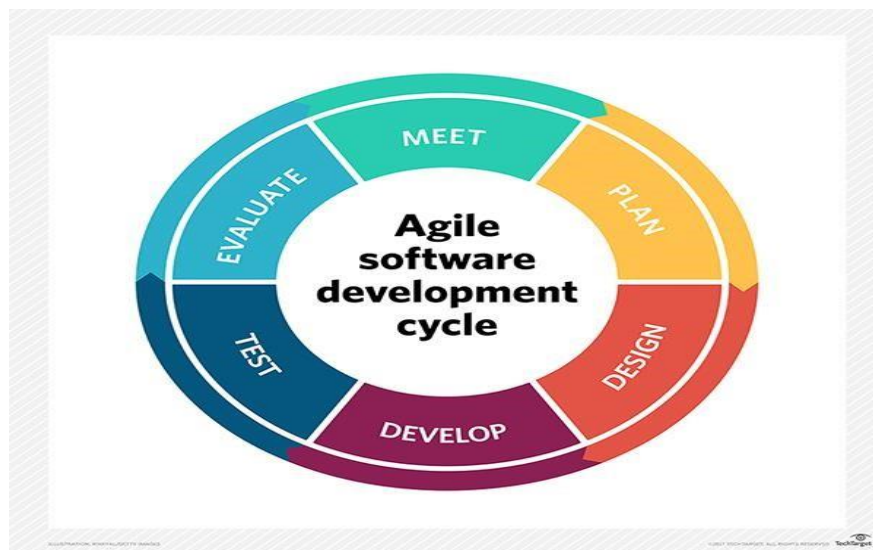
This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

### Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

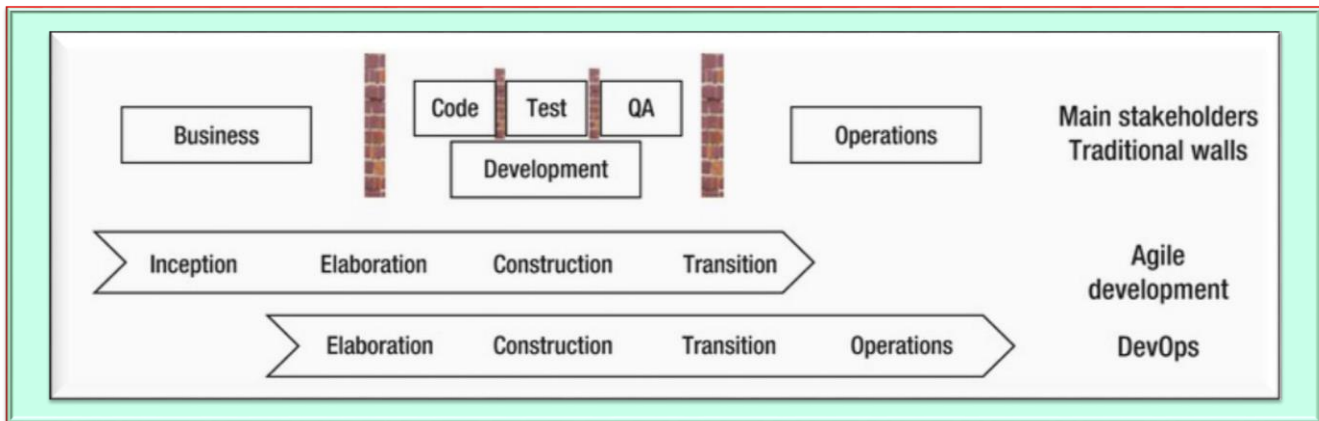
## Agile Methodology

- Developers primarily focus on accelerating the creation of new features by, for instance, adopting Agile methodologies. The Agile movement has brought together programmers, testers, and business representatives.
- Conversely, operations teams are isolated groups that maintain stability and enhance performance by applying practices such as the Information Technology Infrastructure Library (ITIL).
- The principles of Agile methods are focused on defining, building and constructing software.
- DevOps helps development teams increase the frequency of application updates. Agile adoption traditionally omits operations, obstructing the delivery pace.
- DevOps removes this obstruction by applying agile values to the deployment, environment configuration, monitoring and maintenance tasks.



**Figure 4 : Agile Methodology Life Cycle**

1. Inception: In this interval, the vision of the system is developed, a project scope is defined, and the business case is justified.
2. Elaboration: In this interval, requirements are gathered and defined, risk factors are identified, and a system architecture is initialized.
3. Construction: In this interval, the software is delivered to the user.
4. Operations: In this interval, the software is available to the user and is maintained by the operations team.
5. Transition phase is from development to operations.



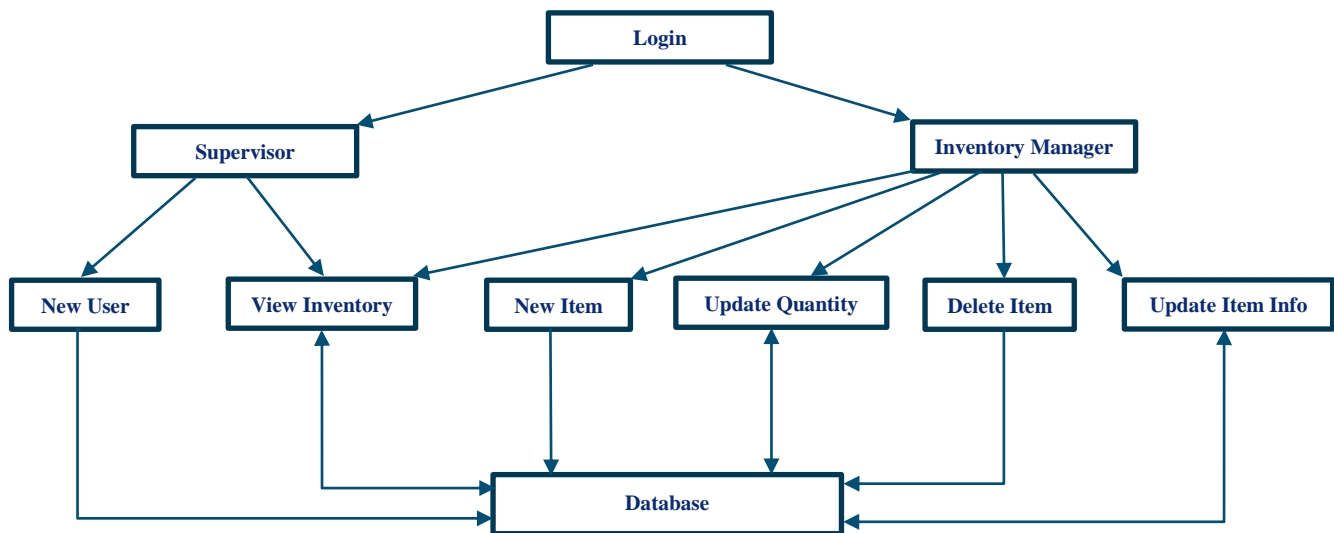
*Figure 5 : Traditional vs Agile vs DevOps*

- Agile breaks the wall between Business and Development team.
- DevOps breaks the wall between Development and Operations team.
- DevOps centers on the concept of sharing: sharing ideas, issues, processes, tools and goals.

## Project Scope

- This android application can be used for tracking the records of available stocks.
- An owner can get availability of a product without being present at warehouse.
- This android application will provide the ease of updating stocks details (Id, Quantity etc.) of different products by inventory manager as and when needed.
- An owner can track details of products in his warehouse in real time.
- An owner has capability to assign roles as Inventory Manager (who updates stock details) and Supervisor (who supervises work of different Inventory Managers).

## Project Architecture



*Figure 6 : Godam Workflow*

1. Login: It is home page of our application software, we use Email-Id and password to authenticate user and the same are used to authorize them as Supervisor or Inventory Manager.
2. Supervisor: A Supervisor has authorization rights to create another Supervisor or an Inventory Manager and view available stocks in the warehouse.
3. Inventory Manager: An Inventory manager has rights to create a new item, update quantity of already existing item, delete an already existing item or update information of the same.
4. New User: A new user can only be created by a Supervisor. A new can either be a Supervisor or an Inventory Manager.
5. View Inventory: It shows real time stock information of products filtered by category and brand.
6. New Item: Only an inventory manager can create a new item. It takes input from inventory manager and update the database with new record.
7. Update Quantity: An inventory manager can update quantity of an existing item using this activity.
8. Delete Item: An inventory manager can delete a particular item from database.
9. Update Item Info: Inventory manager can update/correct information of existing item in the database.

---

## Source Control Management

Source control (or version control) is the practice of tracking and managing changes to code. Source control management (SCM) systems provide a running history of code development and help to resolve conflicts when merging contributions from multiple sources.

Whether we are writing a simple application on our own or collaborating on a large software development project as part of a team, source control is a vital component of the development process. Source code management systems allow us to track our code change, see a revision history for our code, and revert to previous versions of a project when needed. With source code management systems, we can collaborate on code with our team, isolate our work until it is ready, and quickly trouble-shoot issues by identifying who made changes and what the changes were. Source code management systems help streamline the development process and provide a centralized source for all our code.

In our project we have used ‘**Git**’ as SCM tool.

### Git

Git is an open-source distributed source code management system. Git allows us to create a copy of our repository known as a branch. Using this branch, we can then work on our code independently from the stable version of our codebase. Once we are ready with our changes, we can store them as a set of differences, known as a commit. We can pull in commits from other contributors to our repository, push our commits to others, and merge our commits back into the main version of the repository.

Every time we commit, Git takes a snapshot of our work and compares it to previous versions with a viewable operation called a diff. If there's been a change from previous commits, Git stores a new snapshot in the repository.

Git has several key benefits:

- Historical Change Tracking – We can review a graph of how our commits have changed over time, see when and by whom changes were made, and revert to a previous commit if needed. This history makes it easier to identify and fix bugs.
- Work as a Team – We can easily share our code with teammates for review before we commit or merge back to the main working branch. Additionally, the branching and review capabilities enable simultaneous development. Multiple people can work on the same file and resolve differences later.

- Improve Team Speed & Productivity – Git makes it easy for our team to track changes to our code. Now we can focus on writing code instead of spending time tracking and merging different versions across our team. Additionally, Git performs computations and stores our main repository locally, making it quicker on most operations than a centralized VCS.
- Availability and Redundancy – Git is a distributed VCS, meaning there is no single, central place where everything is stored. In a distributed system, there are multiple backups in the event that we need one. This approach also means that we can work offline and commit our changes when we're ready.
- Git Is the Industry Standard – Due to its popularity, Git is supported by many integrated development environments (IDEs) and many popular developer tools including AWS CodeCommit, Jenkins, and Travis. There are many free Git resources available, such as the Git open source page.

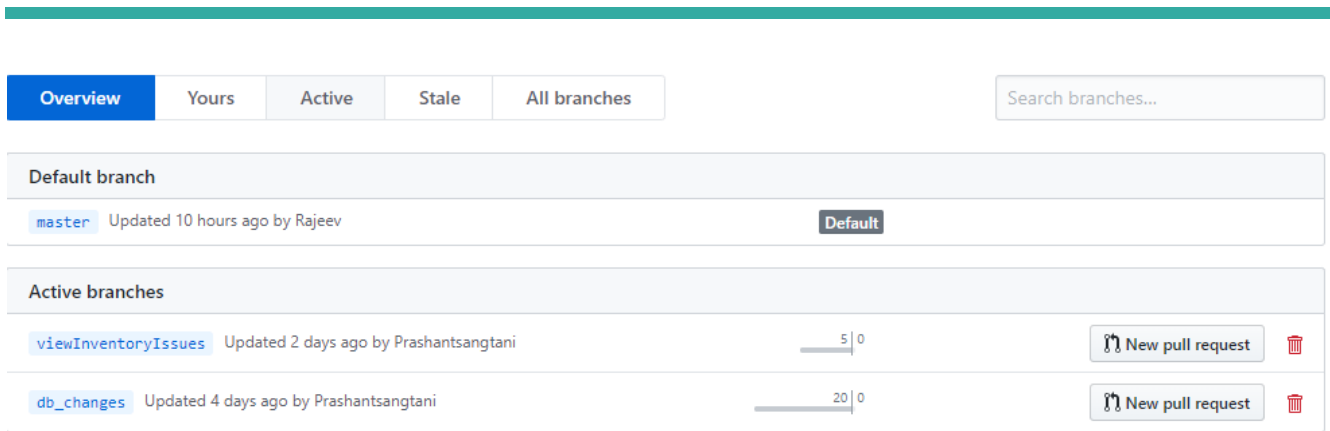
77 commits
3 branches
25 releases
3 contributors

Branch: master
New pull request
Create new file
Upload files
Find File
Clone or download

RajeevPankajShukla added ViewInventory functionality
Latest commit f57412b 10 hours ago

.idea	logo must be changed	11 hours ago
app	added ViewInventory functionality	10 hours ago
gradle/wrapper	initial commit	26 days ago
.gitignore	initial commit	26 days ago
README.md	Update README.md	20 days ago
build.gradle	added firebase authentication:MainActivity.java	24 days ago
gradle.properties	initial commit	26 days ago
gradlew	initial commit	26 days ago
gradlew.bat	initial commit	26 days ago
settings.gradle	initial commit	26 days ago

**Figure 7 : Godam Git Repository**



*Figure 8 : Godam Git Repository Branches*

## Build Tools

Build automation is the act of scripting or automating a wide variety of tasks that software developers do in their day-to-day activities including things like:

- compiling computer source code into binary code
- packaging binary code
- running tests
- deployment to production systems
- creating documentation and/or release notes

Historically, developers used build automation to call compilers and linkers from inside a build script versus attempting to make the compiler calls from the command line. It is simple to use the command line to pass a single source module to a compiler and then to a linker to create the final deployable object. However, when attempting to compile and link many source code modules, in a particular order, using the command line process is not a reasonable solution. The make scripting language offered a better alternative. It allowed a build script to be written to call in a series, the needed compile and link steps to build a software application.

This was the beginning of Build Automation. Its primary focus was on automating the calls to the compilers and linkers. As the build process grew more complex, developers began adding pre and post actions around the calls to the compilers such as a check-out from version control to the copying of deployable objects to a test location. The term "build automation" now includes managing the pre and post compile and link activities as well as the compile and link activities.

In our project, we have use ‘**Gradle**’ as a build tool.



## Gradle

Gradle is an open-source build automation tool focused on flexibility and performance. Gradle build scripts are written using a Groovy or Kotlin DSL.

Gradle is an open-source build automation system that builds upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language (DSL) instead of the XML form used by Apache Maven for declaring the project configuration. Gradle uses a directed acyclic graph ("DAG") to determine the order in which tasks can be run.

Gradle was designed for multi-project builds, which can grow to be quite large. It supports incremental builds by intelligently determining which parts of the build tree are up to date; any task dependent only on those parts does not need to be re-executed.

- Highly customizable — Gradle is modeled in a way that is customizable and extensible in the most fundamental ways.
- Fast — Gradle completes tasks quickly by reusing outputs from previous executions, processing only inputs that changed, and executing tasks in parallel.
- Powerful — Gradle is the official build tool for Android, and comes with support for many popular languages and technologies.

```
Task :app:signingConfigWriterDebug
Task :app:mergeDebugJniLibFolders
Task :app:transformNativeLibsWithMergeJniLibsForDebug
Task :app:processDebugJavaRes
Task :app:transformResourcesWithMergeJavaResForDebug
Task :app:packageDebug
Task :app:assembleDebug

> Task :app:processDebugResources
> Task :app:compileDebugJavaWithJavac
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
> Task :app:compileDebugNdk NO-SOURCE
> Task :app:compileDebugSources
> Task :app:mergeDebugShaders
> Task :app:compileDebugShaders
> Task :app:generateDebugAssets
> Task :app:mergeDebugAssets
> Task :app:mergeExtDexDebug
> Task :app:transformClassesWithDexBuilderForDebug
> Task :app:mergeDexDebug
> Task :app:validateSigningDebug
> Task :app:signingConfigWriterDebug
> Task :app:mergeDebugJniLibFolders
> Task :app:transformNativeLibsWithMergeJniLibsForDebug
> Task :app:processDebugJavaRes NO-SOURCE
> Task :app:transformResourcesWithMergeJavaResForDebug
> Task :app:packageDebug
> Task :app:assembleDebug

BUILD SUCCESSFUL in 54s
29 actionable tasks: 27 executed, 2 up-to-date
Build step 'Invoke Gradle script' changed build result to SUCCESS
Triggering a new build of Test
Finished: SUCCESS
```

Page generated: Apr 23, 2019 12:10:02 PM IST [REST API](#) [Jenkins ver. 2.164.2](#)

**Figure 9 : Jenkins Gradle Build**

---

## Test Tools

Testing is performed by any organization to ensure the quality of the product and continual improvement on the compliance. Testing is the product certification after the product developed and before the customer release.

Testing is a process of executing a program with intent of finding an error or testing is a process used to help to identify the correctness, completeness and quality of developed computer software.

Now days we can get lots of Software Testing Tools in the market. Selection of tools is totally based on the project requirements & commercial (Proprietary/Commercial tools) or free tools (Open Source Tools) we are interested. Off Course, free Testing Tools may have some limitation in the features list of the product, so it's totally based on what are we looking for & is that our requirement fulfills in free version or go for paid Software Testing Tools.

In our project, we have used '**JUnit**' testing framework to test our application software.

### JUnit

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit.

- Every time we make a small or a big modification in the code (in any function), we can make sure that the function is performing well and has not broken any older functionality by executing all JUnit test cases in one go written for that function. So, we write a test case once, and go on using the test case again and again to make sure that- every time software is modified it is working as expected.
- Using JUnit, we can easily create and most importantly manage a rich unit test case suite for the entire software.
- Any new member in the team can easily understand the test cases written and managed using JUnit and consequently contribute towards writing more test cases for developing a robust software.
- JUnit has become a standard for testing in Java programming language and is supported by almost all IDE's e.g. NetBeans, Eclipse etc.

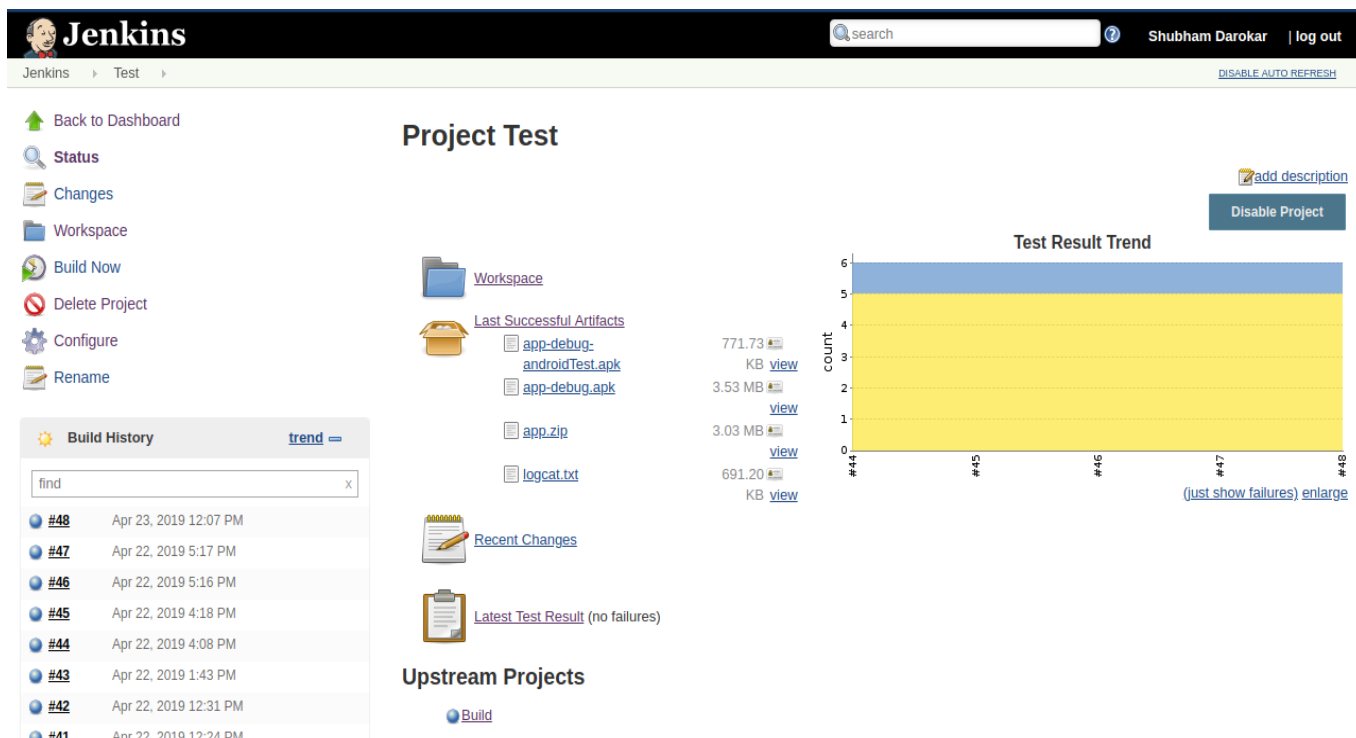
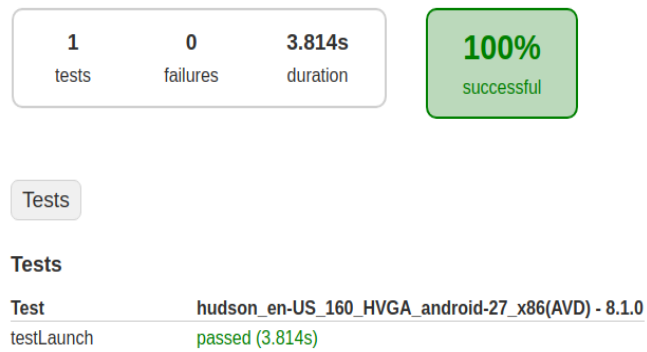


Figure 10 : Jenkins JUnit Tests

## Class com.example.godam.MainActivityTest

all > com.example.godam > MainActivityTest



Generated by Gradle 4.10.1 at 23 Apr, 2019 12:08:59 PM

Figure 11 : Jenkins Test Result

---

## Continuous Integration and Deployment Tools

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

By integrating regularly, we can detect errors quickly, and locate them more easily.

Continuous deployment is a strategy for software releases wherein any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software's users.

Continuous deployment eliminates the human safeguards against unproven code in live software. It should only be implemented when the development and IT teams rigorously adhere to production-ready development practices and thorough testing, and when they apply sophisticated, real-time monitoring in production to discover any issues with new releases.

### Jenkins

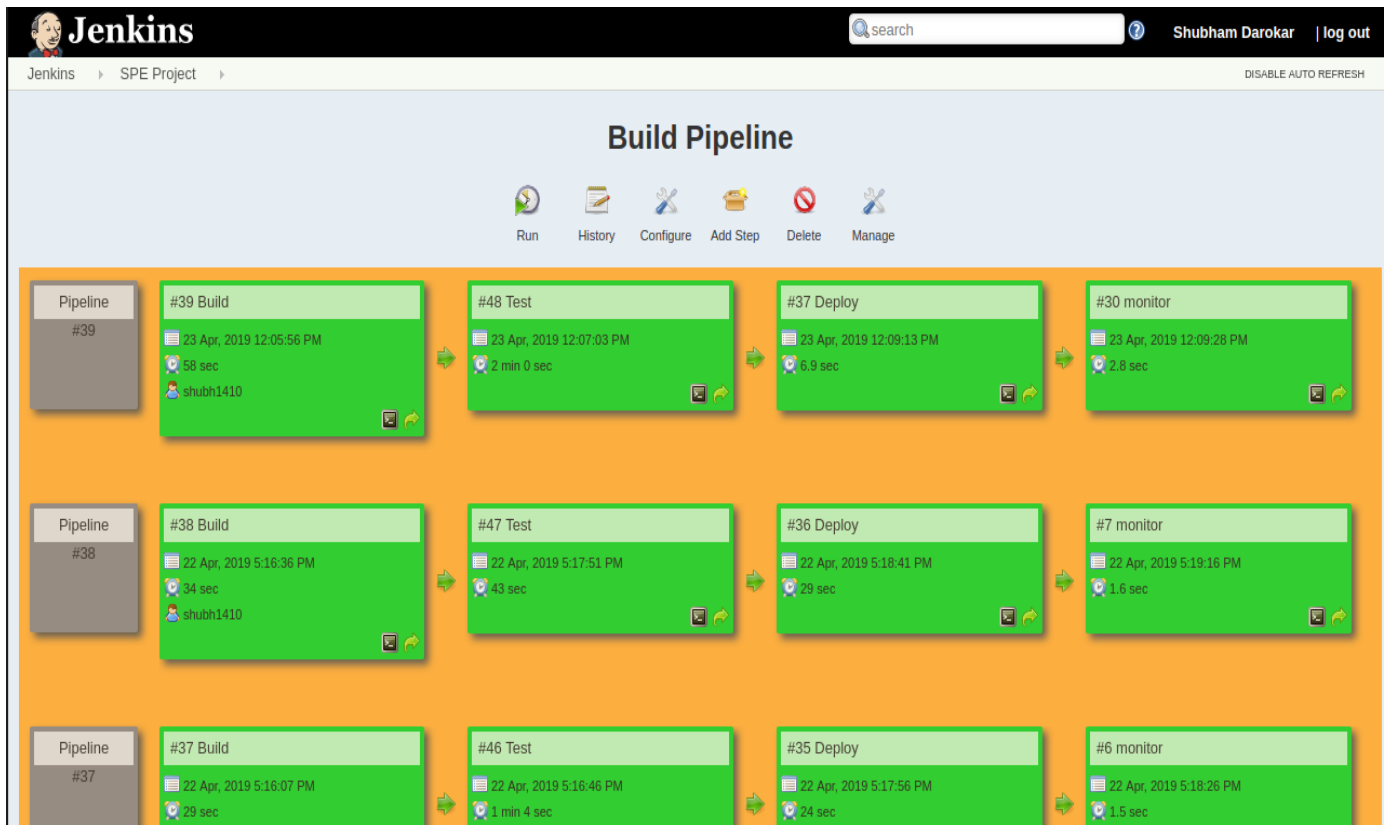
Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

Jenkins is a highly extensible product whose functionality can be extended through the installation of plugins

Advantages of using Jenkins

- It is open source and it is user-friendly, easy to install and does not require additional installations or components.
- Easily Configurable. Jenkins can be easily modified and extended. It deploys code instantly, generates test reports. Jenkins can be configured according to the requirements for continuous integrations and continuous delivery.
- Platform Independent.
- Rich Plugin ecosystem. The extensive pool of plugins makes Jenkins flexible and allows building, deploying and automating across various platforms.
- Easy support. Because it is open source and widely used, there is no shortage of support from large online communities of agile teams.
- Developers write the tests to detect the errors of their code as soon as possible. So, the developers don't waste time on large-scale error-ridden integrations.

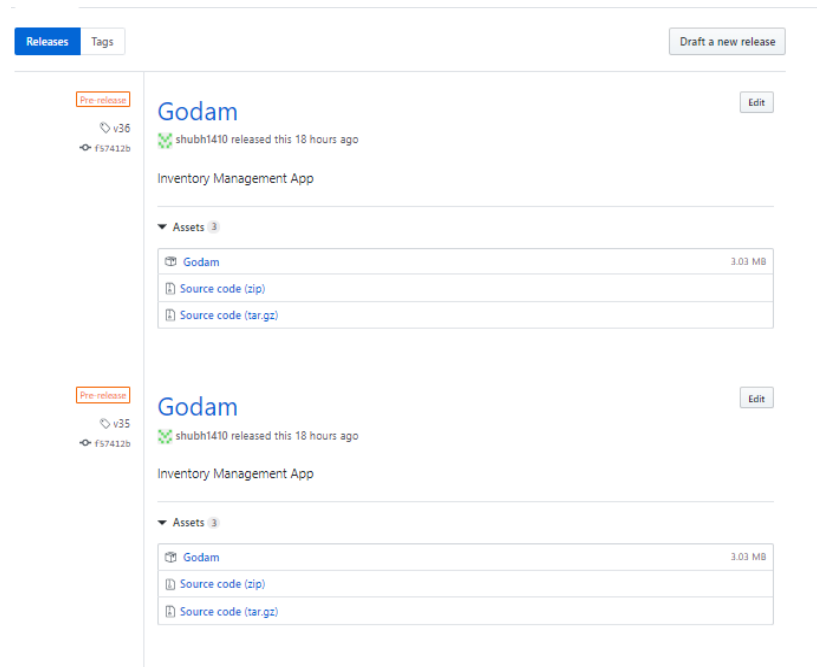
- Issues are detected and resolved almost right away which keeps the software in a state where it can be released at any time safely.
- Most of the integration work is automated. Hence fewer integration issues. This saves both time and money over the lifespan of a project.



**Figure 12 : Jenkins Pipeline**

## GitHub-Release

We can create a release to package software, along with release notes and links to binary files, for other people to use. Releases are based on Git tags, which mark a specific point in our repository's history. Releases are ordered by the date they are created on GitHub.



## Monitoring Tools

Continuous Monitoring is the formal process of defining an agency's IT systems, categorizing each of these systems by the level of risk, application of the controls, continuous monitoring of the applied controls, and the assessment of the effectiveness of these controls against security threats.

Talking about IT, things happen, and changes occur in the blink of an eye. Companies have to continuously work on implementing updated security measures and identify the loopholes in the existing measures which may occur because of some unexpected changes to firmware, software and even hardware.

Continuous monitoring is important because the process is skeptical about potential threats. A good continuous monitoring program is the one that is flexible and features highly reliable, relevant and effective controls to deal with the potential threats.

In our project, we have used ELK Stack for continuous monitoring.

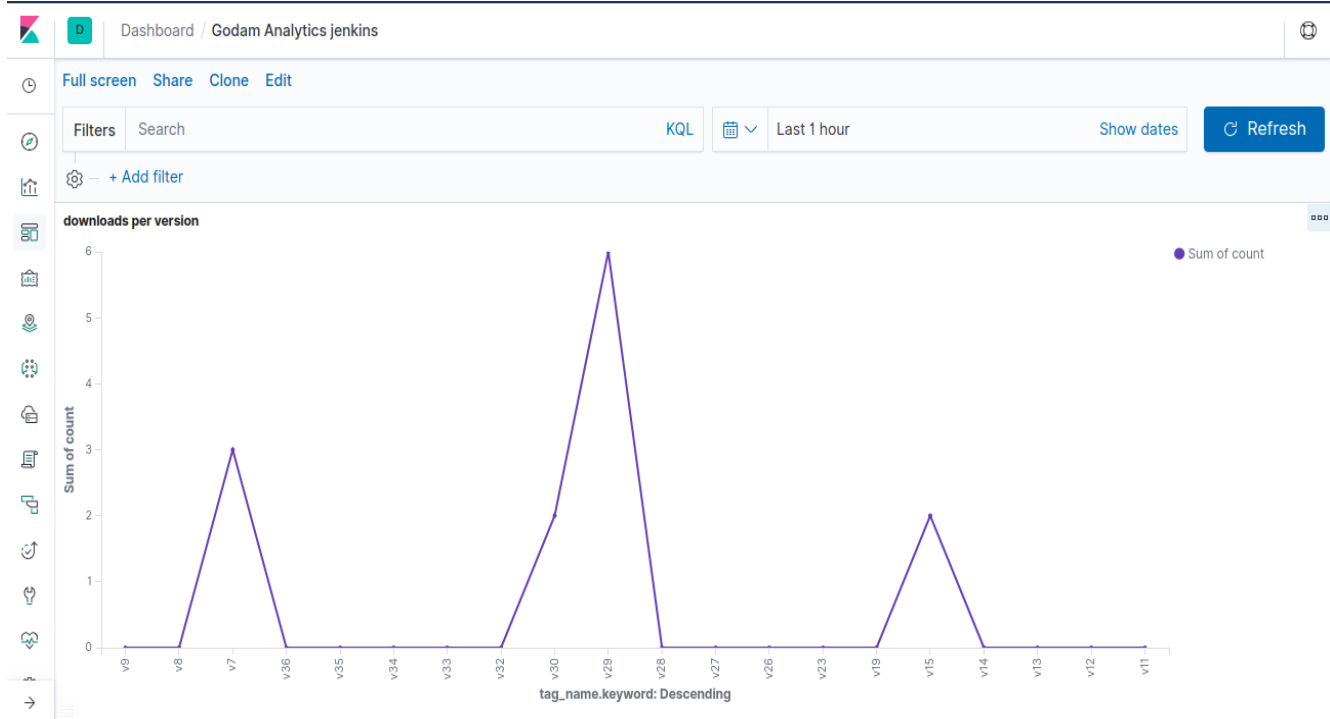
## ELK Stack

"ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana. Elasticsearch is a search and analytics engine. Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch. Kibana lets users visualize data with charts and graphs in Elasticsearch.

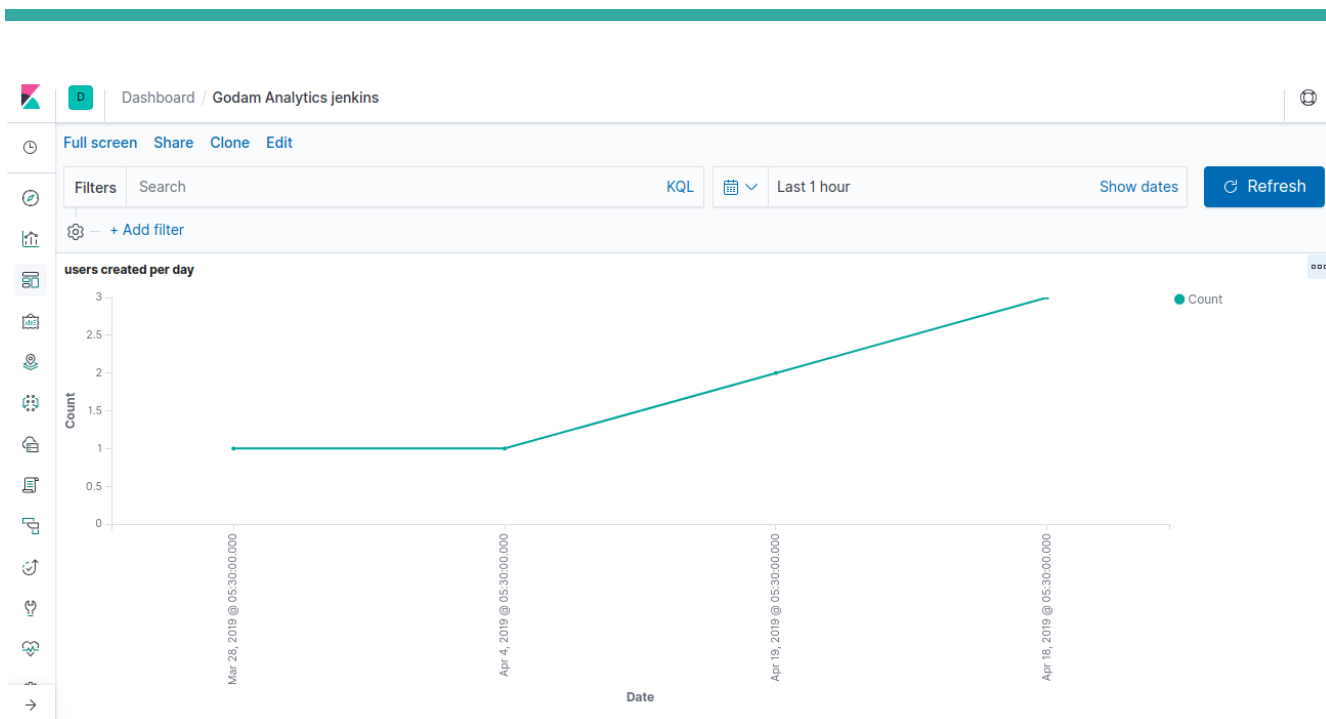
Advantages of using ELK Stack are as follows:

- Not only is it critical for diagnosing and resolving bugs and production issues, but it's increasingly valuable for customer insights.
- Moreover, it helps us in gaining additional metrics about the health and usage of systems gives our team a strong competitive advantage.

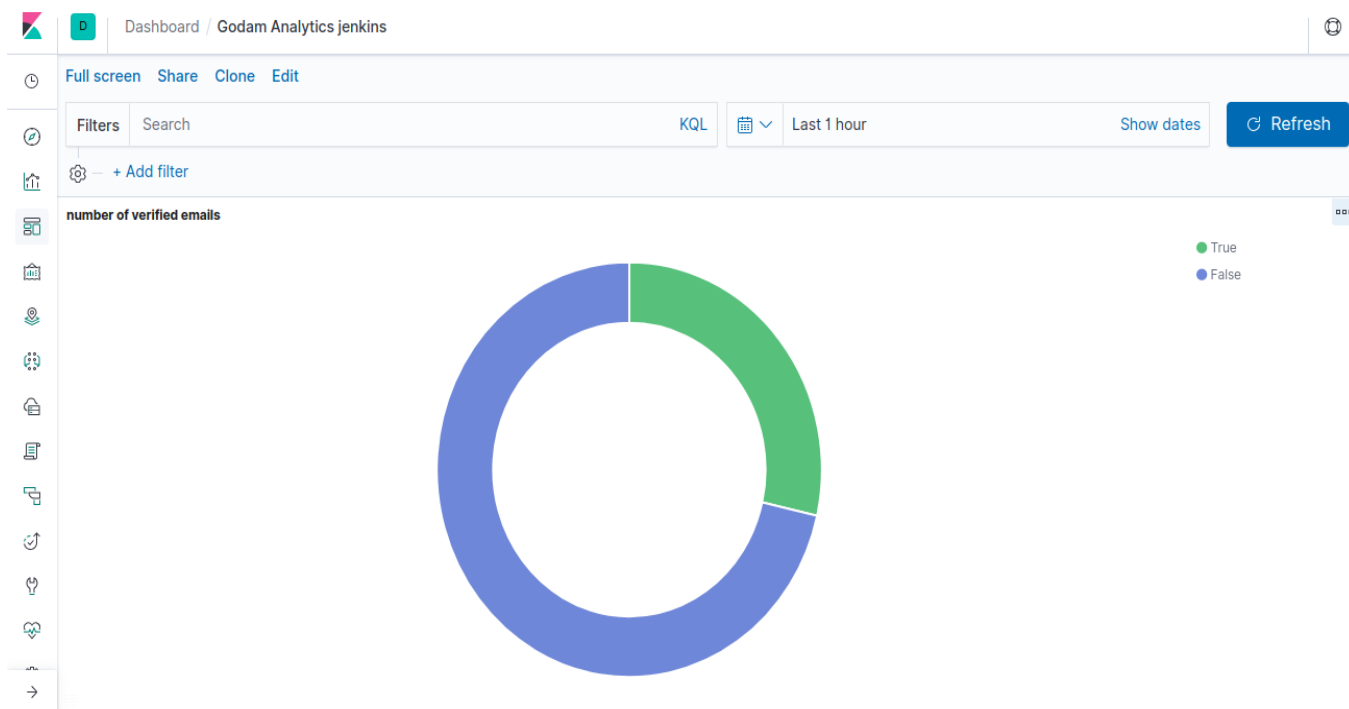
Some of the Kibana Visualization for our project are as follows:



**Figure 13 : Downloads vs Version**



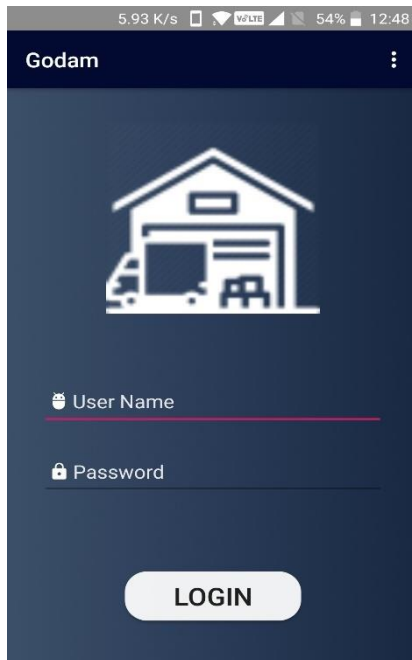
**Figure 14 : Users Created per Day**



**Figure 15 : Number of Verified Emails**




# Results



Godam

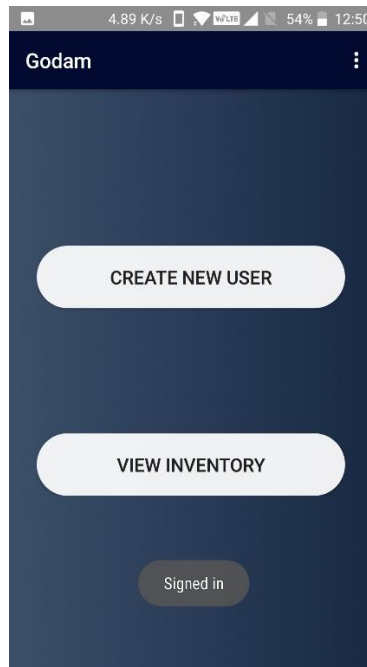
5.93 K/s 54% 12:48



User Name

Password

LOGIN



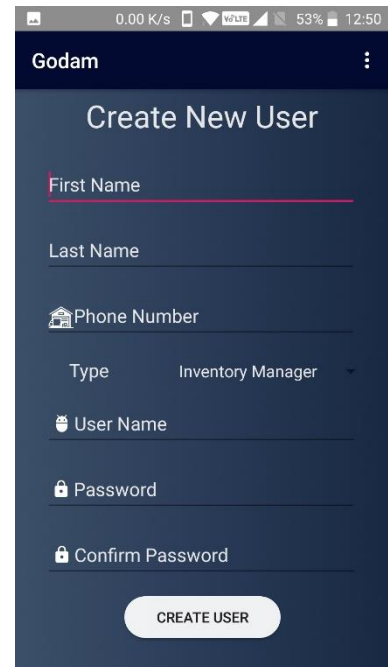
Godam

4.89 K/s 54% 12:50

CREATE NEW USER

VIEW INVENTORY

Signed in



Godam

0.00 K/s 53% 12:50

Create New User

First Name

Last Name

Phone Number

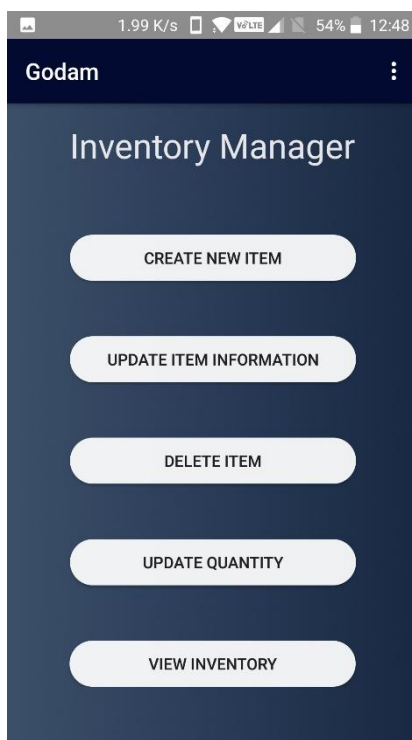
Type Inventory Manager

User Name

Password

Confirm Password

CREATE USER



Godam

1.99 K/s 54% 12:48

Inventory Manager

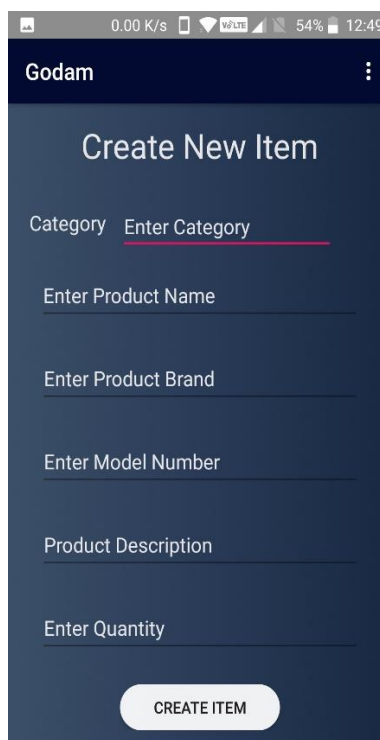
CREATE NEW ITEM

UPDATE ITEM INFORMATION

DELETE ITEM

UPDATE QUANTITY

VIEW INVENTORY



Godam

0.00 K/s 54% 12:49

Create New Item

Category Enter Category

Enter Product Name

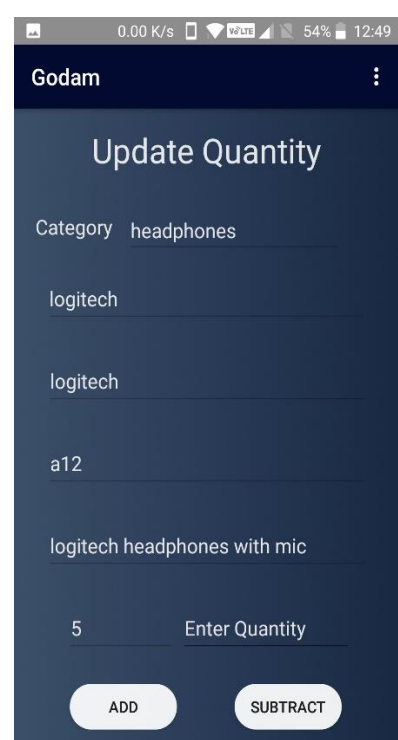
Enter Product Brand

Enter Model Number

Product Description

Enter Quantity

CREATE ITEM



Godam

0.00 K/s 54% 12:49

Update Quantity

Category headphones

logitech

logitech

a12

logitech headphones with mic

5 Enter Quantity

ADD SUBTRACT

---

## Future Work

Current version of Godam application bridges the information gap of owner sitting in shop and stocks in warehouse but same application can be extended to use in shops for billing or to know total worth of goods kept in the warehouse. This android application can also be extended to cross check product list supplied by supplier and received at warehouse.

## Conclusion

The application Godam is working flawlessly for businesses to operate warehouses, where owner keeps the records of available stocks. This project eliminates the paperwork, human faults, manual delay and speed up the process. Godam will have the ability to track available stock in real time. This is simple, fast and intelligent stock management that can be used by anyone who has a smartphone.

We have used various open source tools to incorporate DevOps methodology in our application development life cycle namely Git, Jenkins, Gradle, JUnit framework and ELK Stack which made our development cycle smooth and efficient. Errors were detected in initial stages and resolved in the next iterations decreasing the release time of our application product.

---

# List of Figures

Figure 1 : DevOps Culture .....	4
Figure 2 : DevOps Cycle.....	6
Figure 3 : Software Development Life Cycle .....	9
Figure 4 : Agile Methodology Life Cycle.....	11
Figure 5 : Traditional vs Agile vs DevOps .....	12
Figure 6 : Godam Workflow .....	13
Figure 7 : Godam Git Repository .....	15
Figure 8 : Godam Git Repository Branches .....	16
Figure 9 : Jenkins Gradle Build .....	17
Figure 10 : Jenkins JUnit Tests.....	19
Figure 11 : Jenkins Test Result .....	19
Figure 12 : Jenkins Pipeline .....	21
Figure 13 : Downloads vs Version .....	23
Figure 14 : Users Created per Day .....	24
Figure 15 : Number of Verified Emails .....	24

---

# References

- 1] SDLC Overview - [https://www.tutorialspoint.com/sdlc/sdlc\\_overview.htm](https://www.tutorialspoint.com/sdlc/sdlc_overview.htm)
- 2] <https://www.hexacta.com/2018/05/07/why-devops-in-two-perspectives/>
- 3] <https://jenkins.io/>
- 4] <https://www.guru99.com/junit-tutorial.html>
- 5] <https://developer.android.com/docs>
- 6] Introduction to Software Production Engineering (2018-T2-CS 816 / Software Production Engineering)
- 7] <https://gradle.org/>
- 8] [https://en.wikibooks.org/wiki/Introduction\\_to\\_Software\\_Engineering/Tools/Build\\_Tools](https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Tools/Build_Tools)
- 9] <https://docs.gradle.org/current/userguide/userguide.html>
- 10] <https://en.wikipedia.org/wiki/Gradle>
- 11] <https://jenkins.io/doc/>
- 12] <https://www.elastic.co/elk-stack>