International Institute of Information Technology, Bangalore

Software Production Engineering Final Project Report

Expense Tracker

In the Guidance of Prof. B. Thangaraju
Teaching Assistant: Vishwesh Vinchurkar

Samarth Patel
MT2020036

Shivam Jain
MT2020002

Vaibhavi Tikone
MT2020006

Contents

1. Abstract

In today's busy and expensive life, we are in a great rush to make money. But at the end of the month, we broke off. As we are unknowingly spending money on little and unwanted things. Personal finance management is an important part of people's lives. However, everyone does not have the knowledge or time to manage their finances in a proper manner. And, even if a person has time and knowledge, they do not bother with tracking their expenses on pen and paper as they find it tedious and time-consuming. So, here we have come up with a solution to make one android application as android devices are handy for tracking users' expenses, incomes and give analysis of their expenses on daily basis. We also have stocks feature to add and update stocks for users who invest their savings in stocks.

Our project has two modules:

1. Android application
2. Flask server

Android application gives all the features to add/update/delete incomes, expenses and stocks. And flask server handles the process of fetching live NSE (The National Stock Exchange of India Limited) stocks data.

Expense Tracker app aims to help everyone who are planning to know their expenses and save from it. Here user can define their own categories for expense type like food, clothing, rent and bills where they must enter the money that has been spent and can add some additional information to specify the expense. User can also define expense categories. User will be able to see pie chart of expense. Although this app is focused on new job holders, interns and teenagers, everyone who wants to track their expense can use this app.

GitHub repository: https://github.com/printSamarth/ExpenseTracker

DockerHub repository: https://hub.docker.com/repository/docker/ssvapp/expense-tracker

Appcenter release:
install.appcenter.ms/users/ssvapp/apps/expensetracker/distribution_groups/tester

2. Introduction

2.1 Overview

Expense tracker is an android application for all those who want to manage their expenses. App tracks users' expenses and incomes and gives analysis of their expenses on daily basis. Users can also update their stocks investment, which will be counted towards income. As, users may have income in cash such as pocket money as well as through bank accounts such as salary or stipend; so, in this app user can specify the mode of income and type of expenses such as food, clothes, etc. User can get their expense analysis of the day. He can also specify the duration of the days to generate expense analysis charts and can also get comparison of previous total expense of the same duration he mentioned.

2.2 Features

- Grouping expenses in categories
- Expense analysis: daily, weekly, monthly, yearly.
- comparison analysis with previous expenses.
- Reminder to add the daily expenses at the end of the day.
- Reminder to add the daily expenses at the end of the day.
- Stocks' valuation

2.3 Why Devops?

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). Our main goal is to automate the continuous integration and continuous deployment also known as CI/CD pipeline. Devops tools help to automate this process. We want to add new features as and when required, so called continuous development. Devops tools help us to achieve our goal. Whenever we make changes in our app by adding or improving some features, we want those changes to be reflected in deployed application. For this, Devops tools help us in continuous integration, testing, deployement and monitoring phases.

2.3.1 Devops Features

- Faster, better product delivery
- Faster issue resolution and reduced complexity
- Greater scalability and availability
- More stable operating environments
- Better resource utilization
- Greater automation
- Greater visibility into system outcomes

3. System Configuration

3.1 Operation system

Android operation system with version 4.1 or above

3.2 CPU and RAM

4 core processor and 2 GB RAM preferable

3.3 Language

Java, android framework

Python, flask framework

3.4 Database

Firebase realtime database

3.5 Devops tools

- Source Control Management - GitHub
- Continuous Integration – Jenkins
- Continuous testing - JUnit
- Containerization – Docker
- Continuous deployment – Ansible
- Monitoring - ELK Stack (Elastic Search, Logstash, Kibana)

4. Software Development life cycle

4.1 Installation

4.1.1 Android Studio

To create android app, we need one tool which automates directory structure for various components and serve as editor which supports various types of files for our project. Android studio is official IDE (Integrated Development Environment) for android app development. It is code editor and developer tool which offers many extra features. One can download and install android studio from official website. They have listed steps to follow for 4 OS. One can follow the steps according to their system OS to build the app. Here is the link: https://developer.android.com/studio/install [1]

4.1.2 Java setup

Android Studio requires OpenJDK version 8 or above to be installed to your system. We have used JDK 8. One can follow the following steps to install JDK 8 in their system.

$ sudo apt update
$ sudo apt install openjdk-8-jdk
$ java –version

```
vaibhavi@G3:~$ java -version
openjdk version "1.8.0_282"
OpenJDK Runtime Environment (build 1.8.0_282-8u282-b08-0ubuntu1~18.04-b08)
OpenJDK 64-Bit Server VM (build 25.282-b08, mixed mode)
```

Figure 1 know your current java version

If you have multiple versions of java installed, you can choose java 8 from running below command:

$ sudo update-alternatives --config java

If you have not installed JDK prior and have only JDK 8 by following above commands, then you will get output as shown below:

```
vaibhavi@G3:~$ sudo update-alternatives --config java
There is only one alternative in link group java (providing /usr/bin/java): /usr
/lib/jvm/java-8-openjdk-amd64/jre/bin/java
Nothing to configure.
```

Figure 2 configuration of java version

4.1.3 Python setup

Any Python version 3 or above will work. We have used Python 3.7 version. Our App needs one flask server for stock updates. So, one need to install some additional library along with Python. For installation you can follow these steps:

$ sudo apt update
$ sudo apt-get install python3.7

To install required libraries run below command.

$ pip3 install nsetools flask flask_cors json

Json library is installed with python. But, check whether it is installed for you or not.

```
vaibhavi@G3:~$ conda list | grep -E 'flask|nsetools|json'
flask                     1.1.2                    pypi_0    pypi
flask-cors                3.0.10                   pypi_0    pypi
nsetools                  1.0.11                   pypi_0    pypi
```

Figure 3 required libraries with versions

4.1.4 Jenkins

Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests.
Here we have used Jenkins as an integration tool. Jenkins is a free and open-source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat.
To install jenkins in your localhost, follow below commands:

$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
$ sudo apt-get update
$ sudo apt-get install jenkins

To access jenkins open web browser: <ip address>:8080 (either in local machine or Windows host). Then login as admin and copy the passwd from /var/lib/jenkins/secrets/initialAdminPassword and paste it in the jenkins login page and create user account.

You can run the Jenkins server in one of two ways: either as a stand-alone application, or deployed as a standard web application onto a Java Servlet container or application server such as Tomcat, JBoss, or GlassFish.

4.1.5 Git

Git is a widely used version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files. As a single developer working on our own project, we can keep track of changes made in our source code using git terminal tool. To download command line tool for ubuntu you can follow below steps:

$ sudo apt update
$ sudo apt install git

If you are using windows, you can follow the steps given in the link:
https://phoenixnap.com/kb/how-to-install-git-windows [2]

4.1.6 Docker

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all the parts it needs, such as libraries and other dependencies, and deploy it as one package. Docker is designed to benefit both developers and system administrators, making it a part of many DevOps (developers + operations) toolchains. For installation on your desired OS follow the link: https://docs.docker.com/engine/install/ [3]
As we are using Jenkins for CI, we need to add Jenkins to docker group, so that Jenkins can use docker to build docker image. We can follow below commands on terminal:

$ sudo usermod –aG docker Jenkins
$ sudo less /etc/gshadow | grep jenkins // to verify

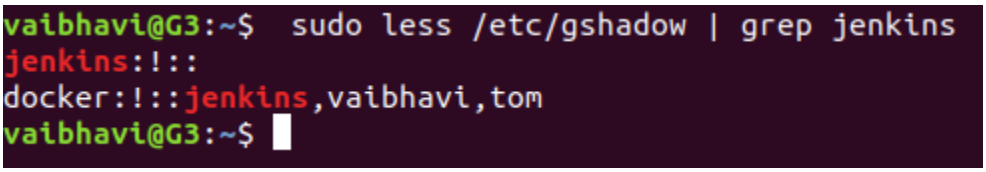

Figure 4 docker group

4.1.7 Ansible

Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. There are two types of hosts in ansible: controller host, and manager host. Controller host is which manages all other hosts where we

want to deploy our project. And manager hosts are those on which we are going to deploy our project.
Ansible connects to the hosts it manages using openssh or winrm and run tasks. In the ansible inventory file we list out manager hosts or group of manager hosts. We can list out all the tasks in the ansible playbook file. One can get the additional information from the official website.
We need to install openssh-server with ansible. So whole process of installation will be done by following below commands:

$ sudo apt install openssh-server
$ sudo apt update
$ sudo apt install ansible
$ ansible --version //to check the ansible version

```
vaibhavi@G3:~$ ansible --version
ansible 2.9.20
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/vaibhavi/.ansible/plugins/modules', u
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.17 (default, Feb 27 2021, 15:10:58) [GCC 7.5.0]
```

Figure 5 Ansible version

Most Ansible modules are written in Python, including the ones central to letting Ansible work. By default, Ansible assumes it can find a **/usr/bin/python** on your system. But, if you want to choose specific interpreter of python, then you can specify path to specific interpreter in inventory file.
We must add manager host to inventory file of ansible as given below. We have created a local user named “tom” on local machine. And then we have added that host to /etc/ansible/hosts file.

```
vaibhavi@G3:~$ cat /etc/ansible/hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers.
localhost ansible_user=tom
## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10
```

Figure 6 specifying the manager nodes

After installing openssh-server one needs to establish connection between controller node and manager nodes. We have local host “tom”, so we need to follow below commands to establish the connection:

On manager host terminal:
$ ssh-key-gen –t rsa
On controller node terminal:
$ ssh-key-gen –t rsa
$ ssh-copy-id tom@localhost

```
samarth@samarth-Inspiron-5447:~$ ssh-copy-id tom@localhost
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out
 any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys
tom@localhost's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'tom@localhost'"
and check to make sure that only the key(s) you wanted were added.
```

Figure 7 One time authentication

If connection has been made successfully then you should be able to login without entering password. We can check by running below command:

$ ssh tom@localhost

```
samarth@samarth-Inspiron-5447:~$ ssh tom@localhost
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.8.0-50-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Pure upstream Kubernetes 1.21, smallest, simplest cluster ops!

     https://microk8s.io/

293 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
tom@samarth-Inspiron-5447:~$
```

Figure 8 remote logging

4.1.8 ELK Stack

The ELK Stack is a collection of three open-source products — Elasticsearch, Logstash, and Kibana.

- E stands for ElasticSearch: used for storing logs

● L stands for LogStash : used for both shipping as well as processing and storing logs
● K stands for Kibana: is a visualization tool (a web interface) which is hosted through Nginx or Apache
ELK Stack is designed to allow users to take data from any source, in any format, and to search, analyze, and visualize that data in real time.

4.1.8.1 ElasticSearch

To download the appropriate version, follow the given link below:
https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.7.0-linux-x86_64.tar.gz

4.1..8.2 LogStash

To download the appropriate version, follow the given link below:
https://artifacts.elastic.co/downloads/logstash/logstash-7.7.0.tar.gz

4.1.8.3 Kibana

To download the appropriate version, follow the given link below:
https://artifacts.elastic.co/downloads/kibana/kibana-7.7.0-linux-x86_64.tar.gz

4.2 Source Control Management

We can either start working with git by initializing directory as git, or we can make one repository on GitHub and clone it, and then we can push our source code to remote repository.
We have used second approach. We have created one public repository on GitHub, and We cloned it. And all team members as collaborators so that they can push changes too.

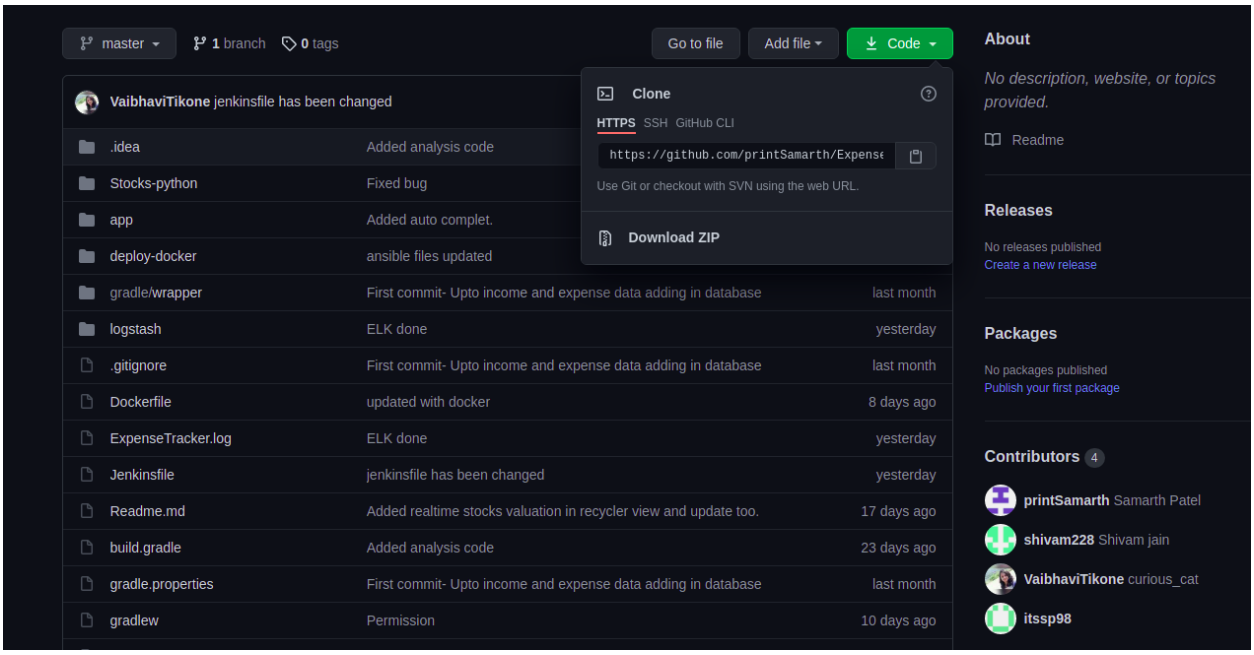

Figure 9 clone the remote repository

To clone the repository go to code button in green color, and in HTTPs copy the given link. After that, go to terminal and follow the below steps to push the code written in the IDE:

$ git clone <paste_link>
$ git add . // to add the changes made in the local repo.
$ git commit –m "message" //to commit the changes
$ git push //finally to push local committed changed to remote repository.

You can create new branch to the repository and add changes over there by following command:

$ git checkout –b <branch-name>

You can anytime check the status of the changes using command as below:

$ git status

Since there may be more than one contributor present for same repository, so if you are working on same branch then you need to pull the changes made by others before adding your changes. You can simply run the pull command as below:

$ git pull // to pull the changes

```
vaibhavi@G3:~/Documents/SPE/final_project/ExpenseTracker$ git pull
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 8 (delta 5), reused 8 (delta 5), pack-reused 0
Unpacking objects: 100% (8/8), done.
From https://github.com/printSamarth/ExpenseTracker
   4830489..4dcd68d  master     -> origin/master
Updating 4830489..4dcd68d
Fast-forward
 Dockerfile                    | 4 ++--
 deploy-docker/deploy-image.yml | 4 ++--
 deploy-docker/inventory       | 2 +-
 logstash/logstash.conf        | 2 +-
 4 files changed, 6 insertions(+), 6 deletions(-)
```

Figure 10 Running git pull command

```
shivam@DESKTOP-U3HNOD9 MINGW64 ~/Documents/ExpenseTracker (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   app/src/main/java/com/example/expensetracker/Registration.java

no changes added to commit (use "git add" and/or "git commit -a")

shivam@DESKTOP-U3HNOD9 MINGW64 ~/Documents/ExpenseTracker (master)
$ git add .

shivam@DESKTOP-U3HNOD9 MINGW64 ~/Documents/ExpenseTracker (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   app/src/main/java/com/example/expensetracker/Registration.java
```

Figure 11 Adding local changes

```
shivam@DESKTOP-U3HNOD9 MINGW64 ~/Documents/ExpenseTracker (master)
$ git commit -m "Registration file modified"
[master 4dde05f] Registration file modified
 1 file changed, 2 insertions(+), 2 deletions(-)

shivam@DESKTOP-U3HNOD9 MINGW64 ~/Documents/ExpenseTracker (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Figure 12 committing changes

```
shivam@DESKTOP-U3HNOD9 MINGW64 ~/Documents/ExpenseTracker (master)
$ git push
Logon failed, use ctrl+c to cancel basic credential prompt.
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/libexec/git-core/git-gui--askpass'
Username for 'https://github.com': shivam228
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/libexec/git-core/git-gui--askpass'
Password for 'https://shivam228@github.com':
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (10/10), 753 bytes | 251.00 KiB/s, done.
Total 10 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To https://github.com/printSamarth/ExpenseTracker.git
   f57ac36..4dde05f  master -> master
```

Figure 13 pushing local changes to remote repository

4.2.1 CI Pipeline

Jenkins support various plugins. To use git within Jenkins to clone the project files, we need to add git plugin to Jenkins. You can manage plugins under manage Jenkins on the dashboard page. Under the available section you can search for git plugin. When you install it, you can verify it under installed section as shown below:

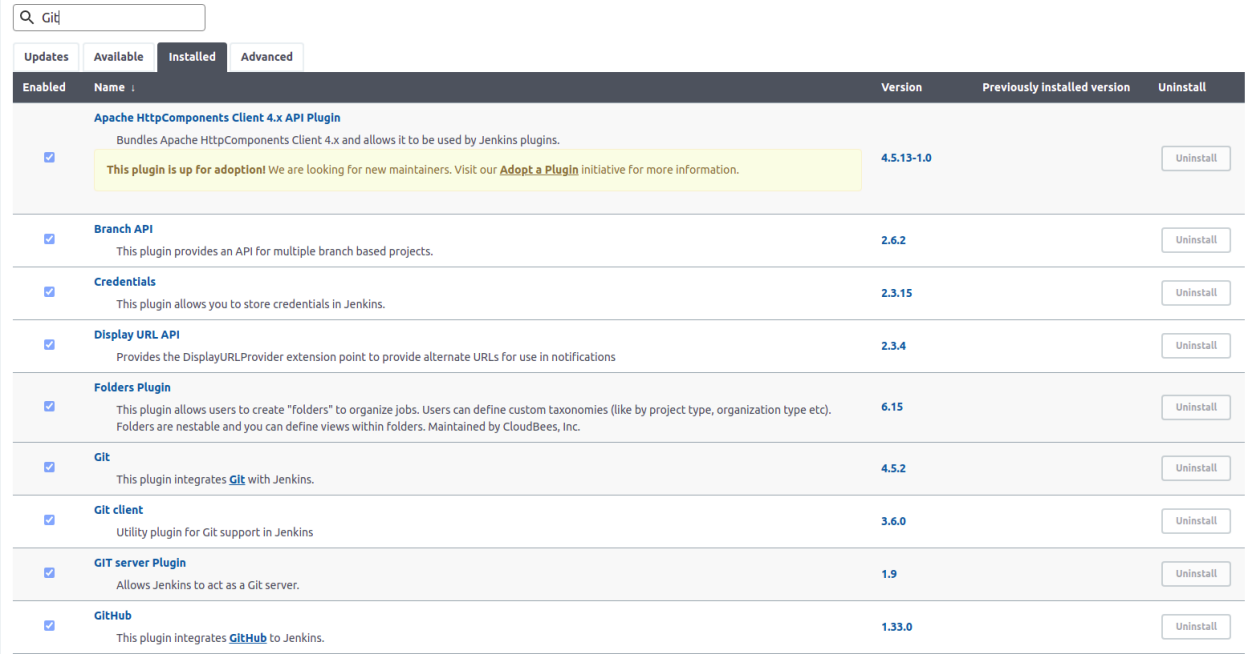

Figure 14 Git plugin in Jenkins

We add credentials (if any) of github repository. Under dashboard, you can find manage credentials under manage jenkins option given. To add the credentials, you can go to (global) under stored scoped to jenkins part.

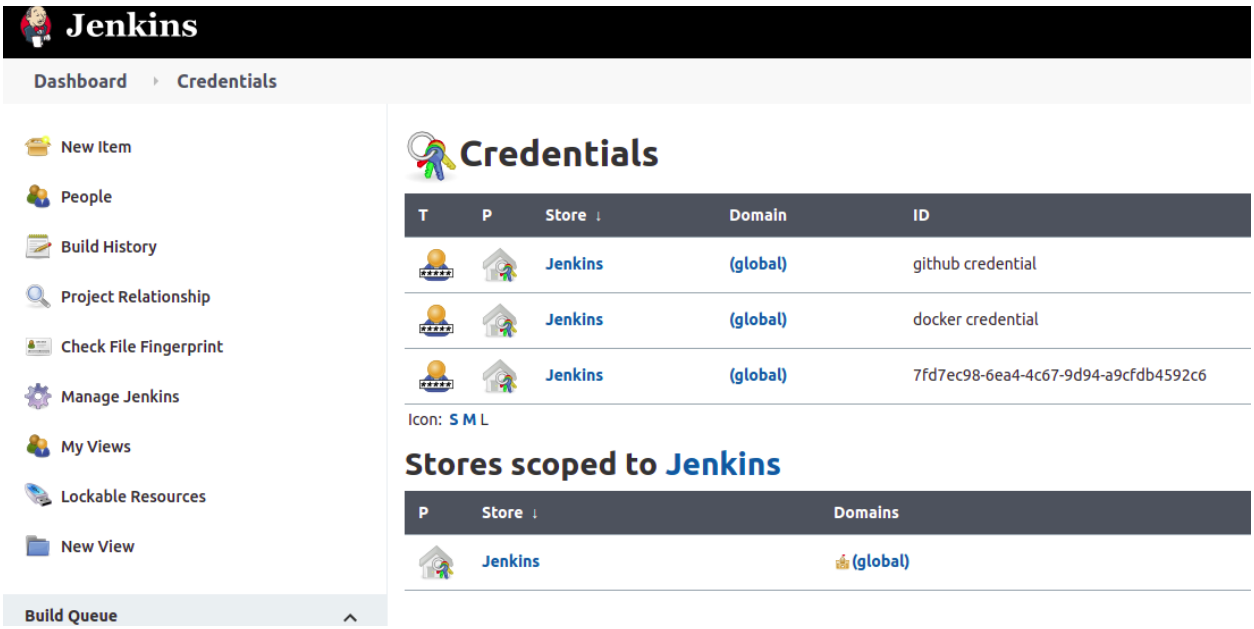

Figure 15 navigation to add credentials

Figure 16 Add Github credentials in jenkins

Now we need to create one pipeline job for our project. To create pipeline, go to new items under dashboard.

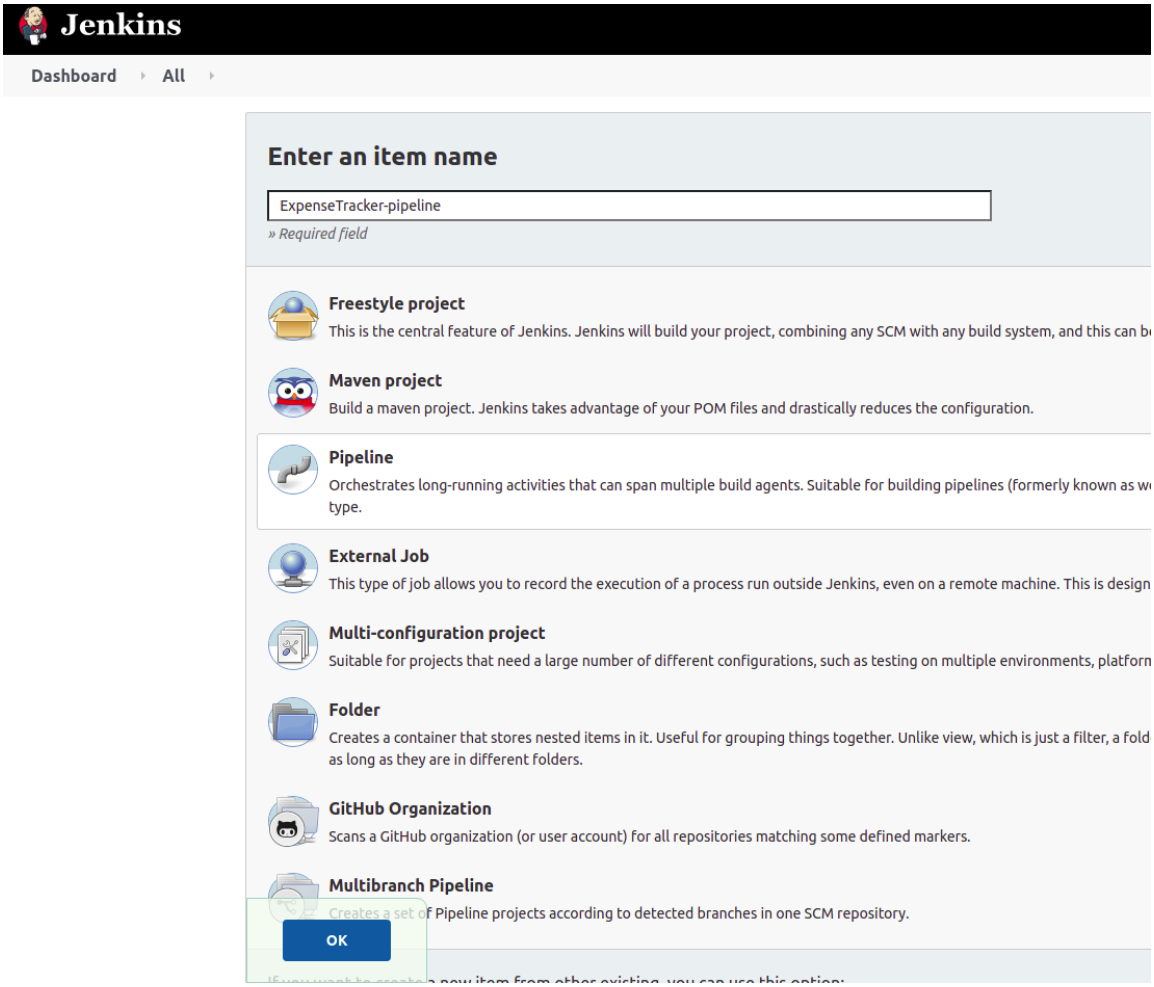

Figure 17 create jenkins pipeline

You can configure pipeline as per your requirement. For instance, if you want to discard old builds to save space, you can specify the number of builds you want to keep at a time as below:

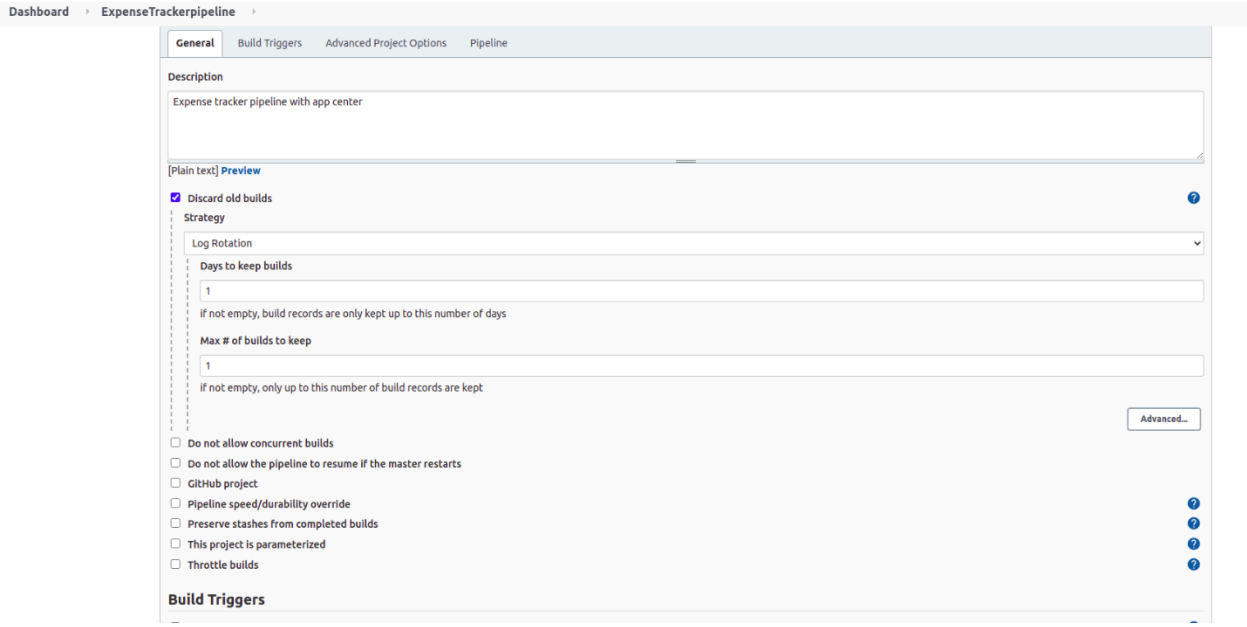

Figure 18 Configuring Jenkins pipeline

To automate the steps of pipeline, Jenkins manages one pipeline script file. We can specify our steps by specifying it according to pipeline script syntax. For fetching code from git remote repository, script would look like as follows:

```
stages {
    stage('Git'){
        steps{
            git 'https://github.com/printSamarth/ExpenseTracker'

        }
    }
```

Figure 19 Jenkins script for git clone

4.3 Build

4.3.1 Gradle

The Android build system compiles app resources and source code, and packages them into APKs that you can test, deploy, sign, and distribute. Android Studio uses Gradle, an advanced build toolkit, to automate and manage the build process, while allowing you to define flexible custom build configurations. Each build configuration can define its own set of code and resources, while reusing the parts common to all versions of your app.

The Android plugin for Gradle works with the build toolkit to provide processes and configurable settings that are specific to building and testing Android applications. Gradle and the Android plugin run independent of Android Studio. This means that you can build your Android apps from within Android Studio, the command line on your machine, or on machines where Android Studio is not installed. You can specify required dependency in the gradle script file as show below:

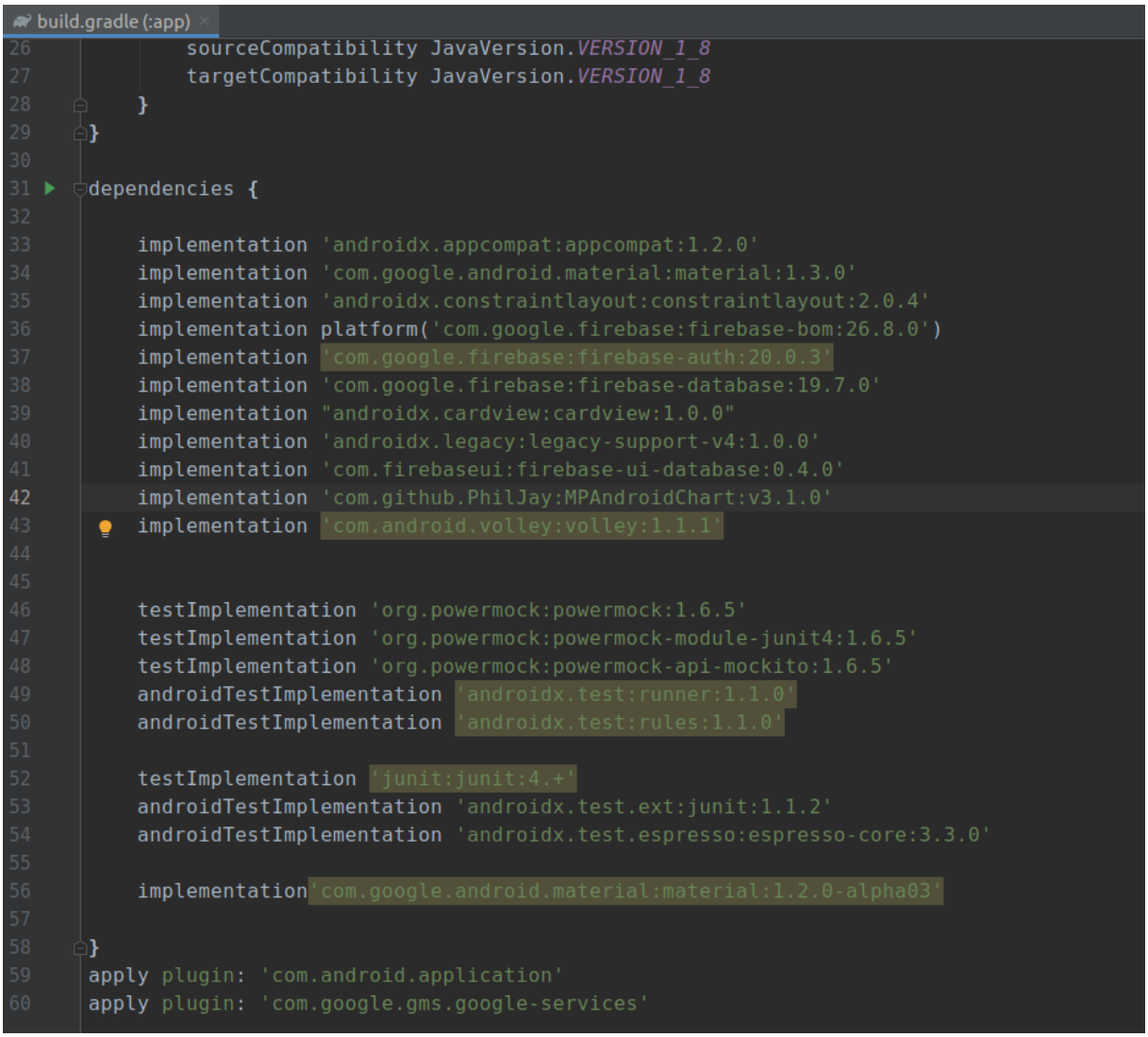

Figure 20 Gradle dependencies

4.3.2 Python

We have installed required libraries and appropriate python version in the installation setup topic. To build the python files, you can use any editor or command line interpreter available in your system. Python files are available under Stocks-python folder.

https://github.com/printSamarth/ExpenseTracker/tree/master/Stocks-python

```
(base) C:\Users\shivam>conda activate base

(base) C:\Users\shivam>python C:\Users\shivam\Documents\ExpenseTracker\Stocks-python\stocks_fetcher.py
 * Serving Flask app "stocks_fetcher" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 741-737-034
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 21 stocks_fetcher server

4.3.3 CI pipeline

To build the project within Jenkins pipeline, we need to install gradle plugin and android emulator plugin. Follow the same steps to install plugins.

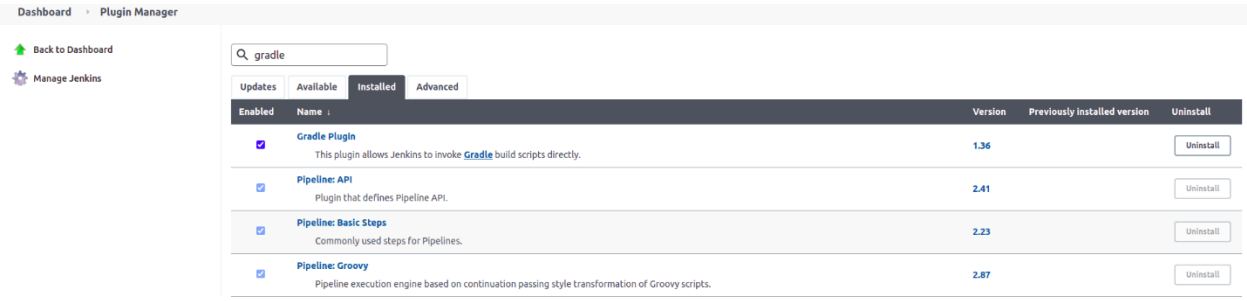

Figure 22 Gradle plugin

We can run below command to build project using gradle:

$ gradle build

Build command of gradle creates buildDir folder which contains the output of the gradle operation.
Sometimes repeated builds result in an unclean build which may be broken due to build artifacts produced by previous builds. To avoid this, we should always run clean command followed by build command. The clean task is defined by the java plugin and it simply removes the buildDir folder, thus cleaning everything including leftovers from previous builds which are no longer relevant.

Jenkins pipeline script for clean and build command is shown below:

```
14          stage('Clean') {
15              steps {
16                  sh './gradlew --refresh-dependencies clean'
17              }
18          }
19          stage('Build'){
20              steps{
21                  sh './gradlew assembleDebug'
22                  archiveArtifacts '**/*.apk'
23              }
24          }
```

Figure 23 Jenkins pipeline script

4.4 Test

Jenkins provides us with continuous integration which includes building and doing unit testing, so every time we push code to github it integrates all the different modules of the project and tests their proper functioning. As we have created android app, we have used Espresso framework as testing tool. The Espresso testing framework, provides APIs for writing UI tests to simulate user interactions within a single target app. You can check these test classes under test folder. The direct link is given below:

https://github.com/printSamarth/ExpenseTracker/tree/master/app/src/androidTest/java/com/example/expensetracker

For instance, test case for main activity module is as shown below:

```java
package com.example.expensetracker;

import androidx.test.core.app.ActivityScenario;
import androidx.test.internal.runner.junit4.AndroidJUnit4ClassRunner;

import org.junit.Test;
import org.junit.runner.RunWith;

import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.action.ViewActions.click;
import static androidx.test.espresso.assertion.ViewAssertions.matches;
import static androidx.test.espresso.matcher.ViewMatchers.isDisplayed;
import static androidx.test.espresso.matcher.ViewMatchers.withId;
import static androidx.test.espresso.matcher.ViewMatchers.withText;
import static org.junit.Assert.*;


@RunWith(AndroidJUnit4ClassRunner.class)
public class MainActivityTest {
    @Test
    public void testActivityInView() {
        ActivityScenario.launch(MainActivity.class);
        onView(withId(R.id.main_login)).check(matches(isDisplayed()));
    }

    @Test
    public void testClickToRegisterButton(){
        ActivityScenario.launch(MainActivity.class);
        onView(withId(R.id.signup_registration)).perform(click());
        onView(withId(R.id.registration)).check(matches(isDisplayed()));
    }
```

Figure 24 Code for main activity test

4.4.1 CI pipeline

In the Jenkins pipeline we write a script to invoke test cases written in our project. Jenkins pipeline is as shown below:

```
25          stage('Install for testing'){
26              steps{
27                  sh './gradlew installDebug'
28              }
29          }
30          stage('Testing'){
31              steps{
32                  sh './gradlew connectedDebugAndroidTest'
33              }
34          }
```

Figure 25 Jenkins pipeline script for testing

Make sure, that while running testing pipeline, your android device is connected to the laptop/PC.

4.5 Docker artifact

We can create docker image and run it in local device using command line docker tool. But this image is only accessible to local user. If we want to share this image with our team-mates, then we need to use Docker Hub.
Docker Hub is a repository registry service provided by Docker Inc. It allows us to pull and push docker images to and from Docker Hub. We can treat this as a GitHub, where we fetch and push our source code, but in the case of Docker Hub, we download or publish our container images.
We are using flask server as service in our project. So, we build one image to deploy flask files and run the server from that image. We need to first create one repository on Docker Hub. For this, one need to create an account on Docker Hub. After that we build the image followed by pushing that image to Docker Hub.

4.5.1 CI pipeline

Jenkins supports docker, too. We need to install Docker plugins for that. And we also need to provide docker credentials in manage credentials section, as we did for Git Hub.

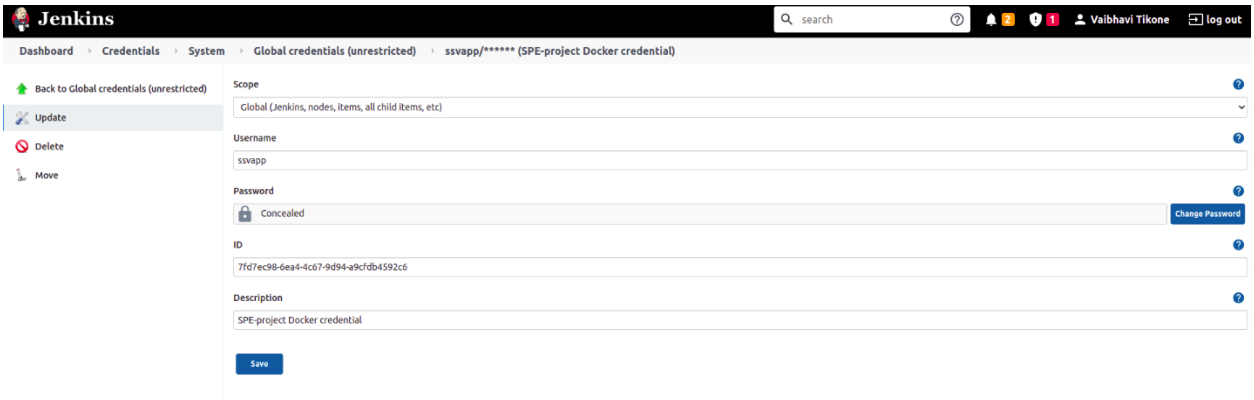

Figure 26 Docker Hub credentials

To build the image we first need to create one Dockerfile in our project folder. Dockerfile has its own syntax that we must follow. We specify the command that we want to run in an image. So, the Dockerfile for our project is as shown below:

```
1   FROM ubuntu:18.04
2
3   MAINTAINER ssvapp speproject039@gmail.com
4
5   RUN apt-get update -y && \
6       apt-get install -y python3-pip python3-dev
7
8   COPY ./requirements.txt /requirements.txt
9
10  WORKDIR /
11
12  RUN pip3 install -r requirements.txt
13
14  COPY . /
15
16  ENTRYPOINT [ "python3" ]
17
18  CMD [ "Stocks-python/stocks_fetcher.py" ]
```

Figure 27 Dockerfile

To build the docker image and push it to Docker Hub remote repository, we will specify steps in Jenkins pipeline script.

```
35          stage('Docker build to Image') {
36                  steps {
37                      script{
38                              imageName = docker.build "ssvapp/expense-tracker:latest"
39                      }
40                  }
41              }
42              stage('Push Docker Image to DockerHub') {
43                  steps {
44                      script{
45                              docker.withRegistry('','7fd7ec98-6ea4-4c67-9d94-a9cfdb4592c6')
46                              {
47                                  imageName.push()
48                              }
49                      }
50                  }
51              }
```

Figure 28 Jenkins pipeline script

After you build this pipeline successfully, you can check whether this image is pushed to remote repository or not on Docker Hub.

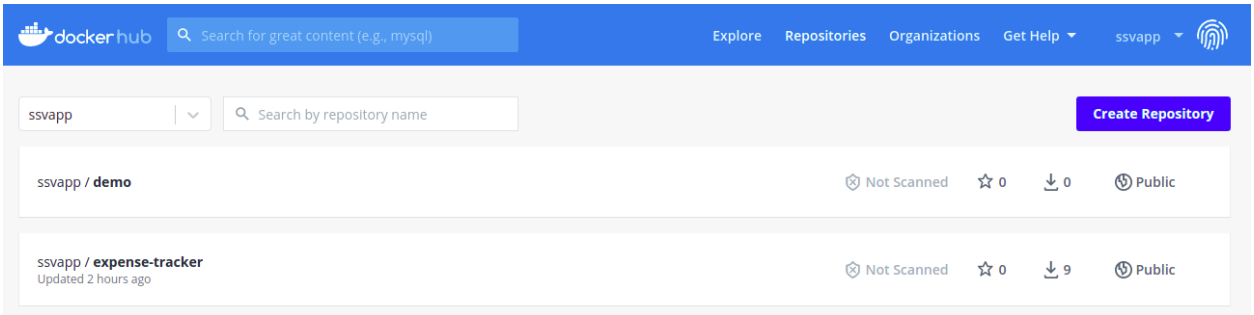

Figure 29 Docker Hub repository

From Docker Hub, anyone can download the image and run in their local host if you make repository public.

4.6 Deploy

We have two modules of our project, and we need to deploy both separately. One is our android application itself, and one is flask server for fetching live stock data. We have used Ansible to deploy flask server using docker image, and visual studio app center to deploy our android application.

Visual Studio App Center is an integrated mobile development lifecycle solution for iOS, Android, Windows and macOS apps. It brings together multiple services commonly used by mobile developers, including build, test, distribute, monitoring, diagnostics, etc., into one single integrated cloud solution.

We have created one docker image of flask server and pushed it to our Docker Hub repository. We have created one separate local host “tom” which will run this server for us. So, we need to pull the docker image, and need to run on local host.

Ansible manages one inventory file to specify manager nodes, and one yml file which acts as task notebook, where we can specify tasks, that we want to run on manager nodes from controller node. We have created these two files under deploy-docker folder. Direct Github link: https://github.com/printSamarth/ExpenseTracker/tree/master/deploy-docker

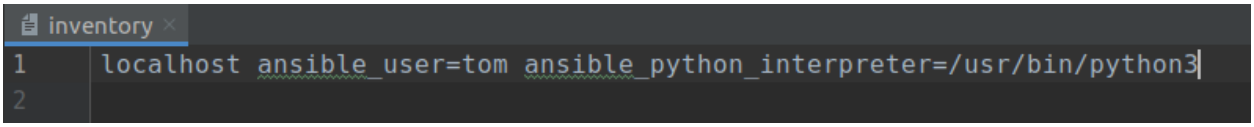

Figure 30 Inventoryfile

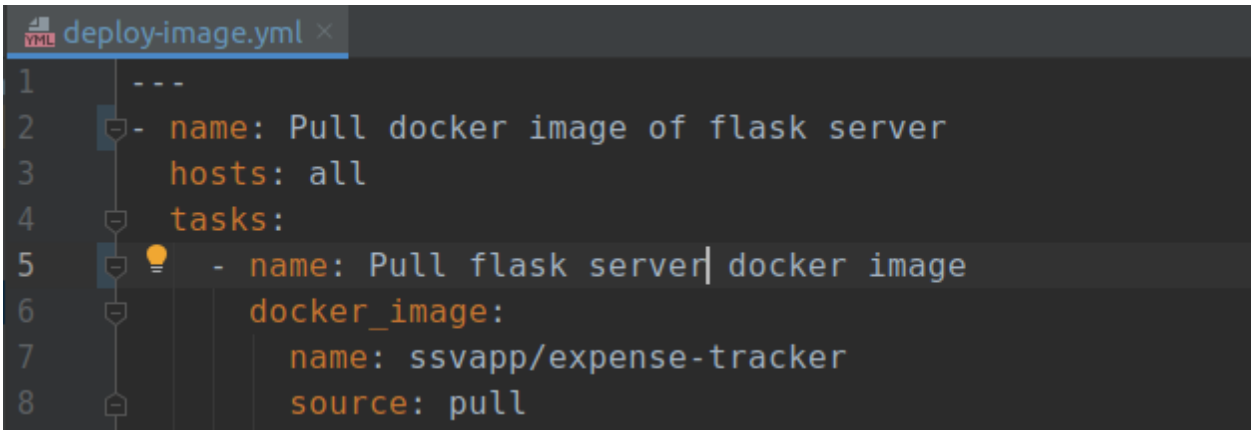

Figure 31 Ansible notebook

To deploy our android application, we need to create one account on app center and need to create one app. After login you can find add app option under all apps.

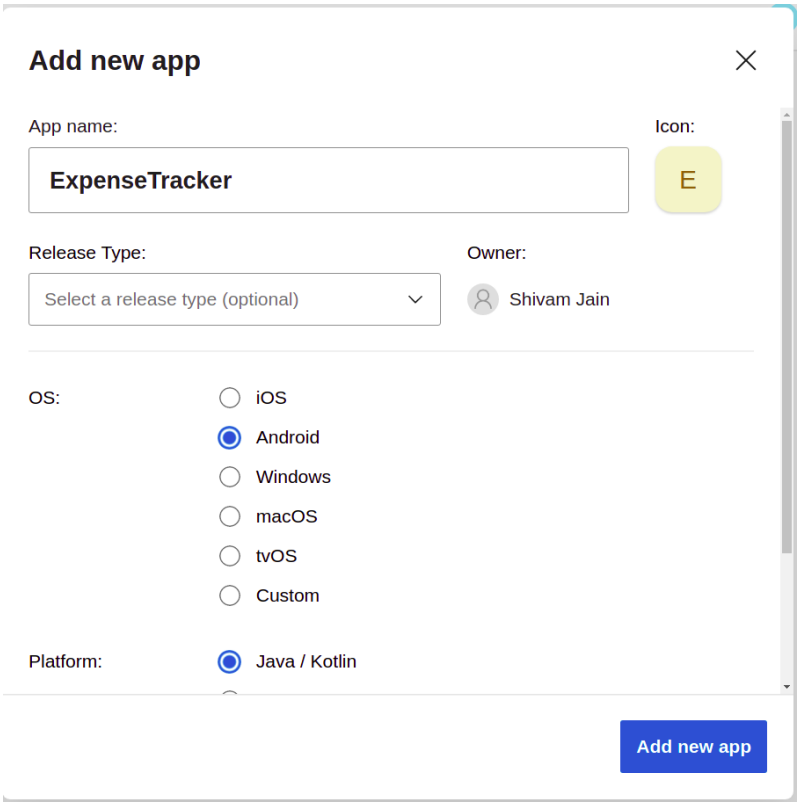

Figure 32 Add new app

After that we need to create one group under distribute option for this application. We have created one tester group.

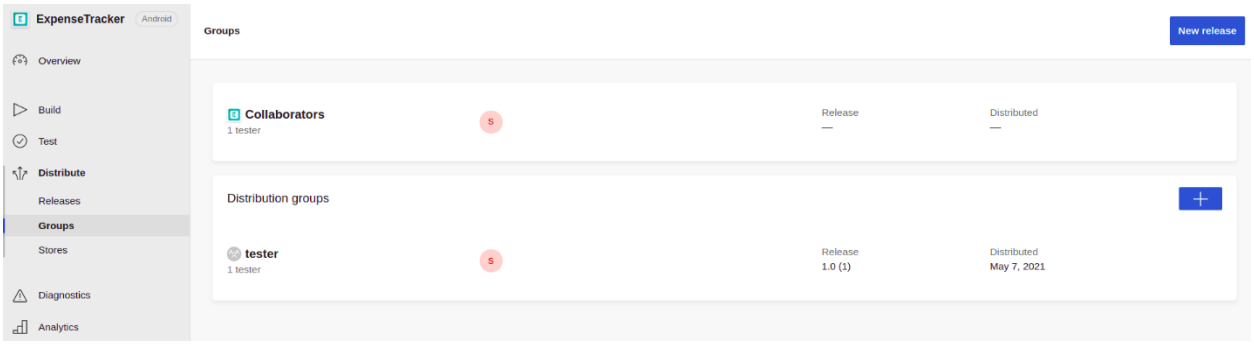

Figure 33 Add group

Now, we need to generate one API token to ditribute this application. You can find this option under your account setting.

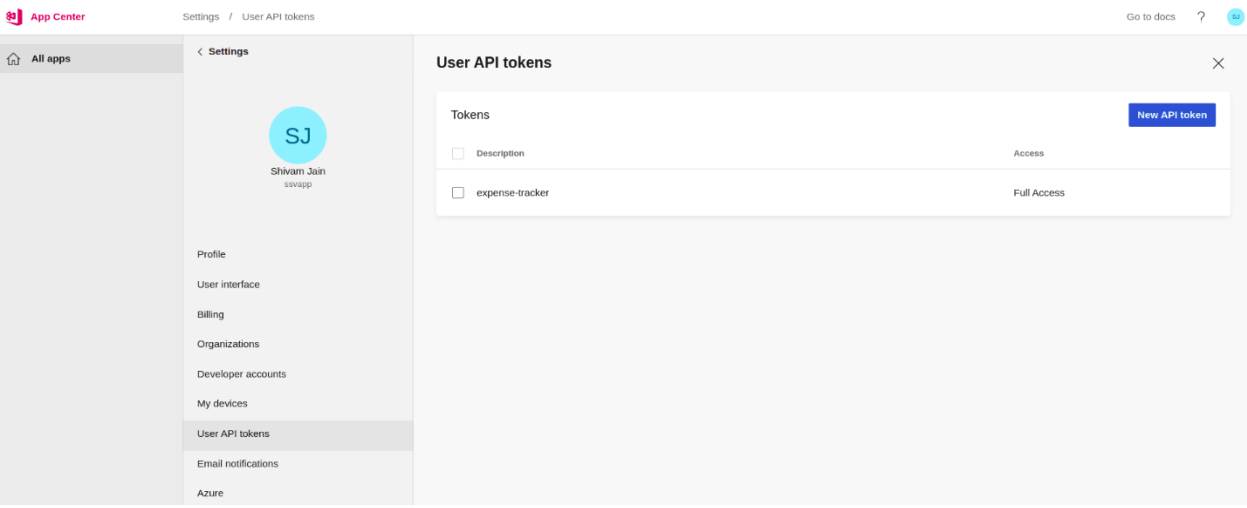

Figure 34 Add API token

You need to copy this api token to somewhere for further use, as it is one time seen.

4.6.1 CD Pipeline

Jenkins supports app center, too. We need to install app center plugin for that. Follow the same steps to install app center plugin in jenkins.

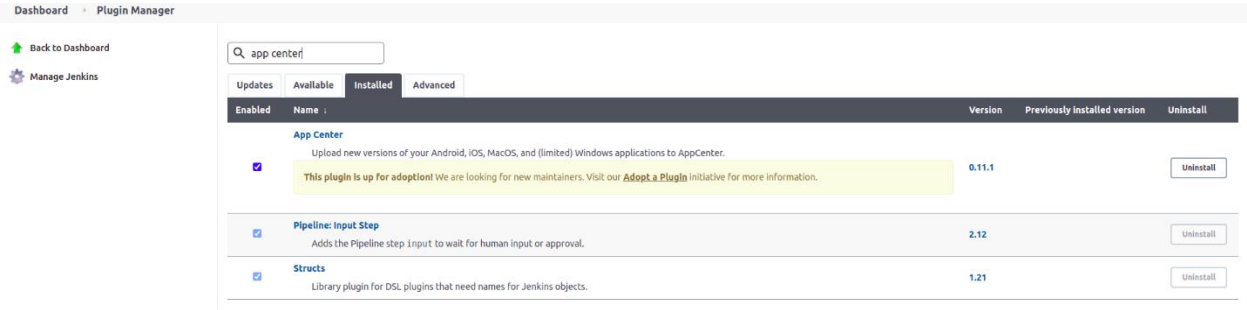

Figure 35 Jenkins app center plugin

Now, we need to write pipeline script for ansible to pull image, and for app center to distribute our app. We need to paste API token, and need to specify owner name, app name and group that we specified in the app center.

```
        stage('Pull Docker Image') {
                        steps {
                           ansiblePlaybook becomeUser: null, colorized: true, disableHostKeyChecking: true,
                           installation: 'Ansible', inventory: 'deploy-docker/inventory',
                           playbook: 'deploy-docker/deploy-image.yml', sudoUser: null
                        }
                    }
        stage('DISTRIBUTE') {
            steps {
                appCenter apiToken: '92a01f4cb9c3bb7a57de2984bde44c67b00f5979',
                        ownerName: 'ssvapp',
                        appName: 'ExpenseTracker',
                        pathToApp: '**/app-debug.apk',
                        distributionGroups: 'tester'
            }
        }
```

Figure 36 Jenkins pipeline script

4.7 Monitor

We have used ELK stack for continuous monitoring our project. Firstly, we have written log statements for each activity of our android app. The Logcat window in Android Studio displays system messages, such as when a garbage collection occurs, and messages that you added to your app with the Log class. It displays messages in real time and keeps a history so you can view older messages. The Log class allows you to create log messages that appear in logcat.

DashBoardFragment.java | Registration.java

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_registration);
    Log.i(TAG, msg: "Inside on create method");
    mEmail = findViewById(R.id.email_registration);
    mPass = findViewById(R.id.password_registration);
    reg_btn = findViewById(R.id.button_registration);
    sign_in = findViewById(R.id.signin_registration);
    try {
        mAuth = FirebaseAuth.getInstance();
    }
    catch (Exception e){
        Log.e(TAG, e.toString());
    }
    Log.i(TAG, msg: "initiating firebase instance");
    mDialog = new ProgressDialog( context: this);
    reg_btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Log.i(TAG, msg: "Registration..");
            String email = mEmail.getText().toString().trim();
            String pass = mPass.getText().toString().trim();
            Log.i(TAG, msg: "Username collected.");
            Log.i(TAG, msg: "Password collected.");
            if(TextUtils.isEmpty(email)){
                Log.e(TAG, msg: "Email is Required");
                Toast.makeText(getApplicationContext(), text: "Email Required", Toast.LENGTH_SHORT).show();
                return;
            }
            if(TextUtils.isEmpty(pass)){
                Log.e(TAG, msg: "Password is required");
                Toast.makeText(getApplicationContext(), text: "Password Required", Toast.LENGTH_SHORT).show();
                return;
            }
            mDialog.setMessage("Processing..");
            Log.i(TAG, msg: "Processing..");
```

Figure 37 Log statements in registration activity

The log message format is :

date time PID-TID/package priority/tag: message

We can check this log statements in logcat terminal.

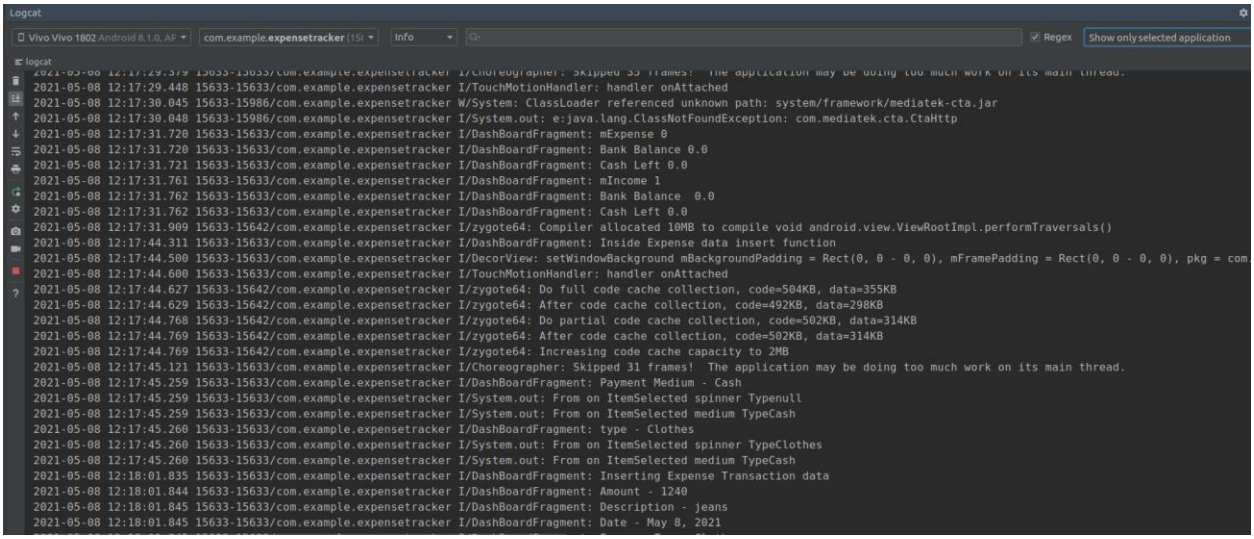

Figure 38 Logcat output

As logcat also shows system generated statements, we can use adb terminal command to get the user provided log statements. To install adb on ubuntu use below command:

$ sudo apt update

$ sudo apt-get install android-tools-adb

After installing adb tool, connect your android device to the laptop/PC. To check whether adb tool has detected your android device or not, run below command:

$ adb devices

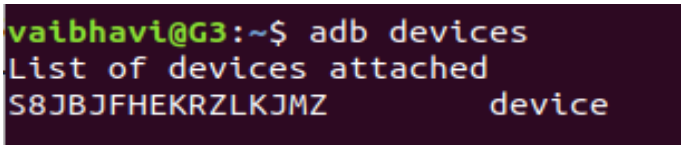

Figure 39 connected devices list

To generate log statements for each of your activities present in your application, run below command:

$ adb logcat MainActivity:D *:S

```
vaibhavi@G3:~$ adb logcat MainActivity:D *:S
--------- beginning of system
--------- beginning of main
--------- beginning of crash
05-08 12:34:37.863 19846 19846 I MainActivity: Inside on create Method
05-08 12:34:45.032 19846 19846 I MainActivity: Username collected
05-08 12:34:45.032 19846 19846 I MainActivity: Password collected
05-08 12:34:45.032 19846 19846 I MainActivity: Validating username and password
05-08 12:34:46.517 19846 19846 I MainActivity: Login Successful Moving to Home Activity
^C
vaibhavi@G3:~$ adb logcat AnalysisFragment:D *:S
--------- beginning of system
--------- beginning of main
05-08 12:33:43.615 19180 19180 I AnalysisFragment: Inside oncreate method
05-08 12:33:43.722 19180 19180 I AnalysisFragment: calling load data function
05-08 12:33:43.723 19180 19180 I AnalysisFragment: Inside load data function
05-08 12:33:43.724 19180 19180 I AnalysisFragment: Maximum spending is done on - Clothes
05-08 12:34:33.827 19180 19180 I AnalysisFragment: Inside oncreate method
05-08 12:34:33.915 19180 19180 I AnalysisFragment: calling load data function
05-08 12:34:33.915 19180 19180 I AnalysisFragment: Inside load data function
05-08 12:34:33.915 19180 19180 I AnalysisFragment: Maximum spending is done on - Clothes
```

Figure 40 Log statements

Make sure that while running adb command your android device is connected to your laptop/pc. You can specify name of your activity for which you want to generate log. For instance, we have specified AnalysisFragment activity. Name should be same as java file. For reference, see the below image:

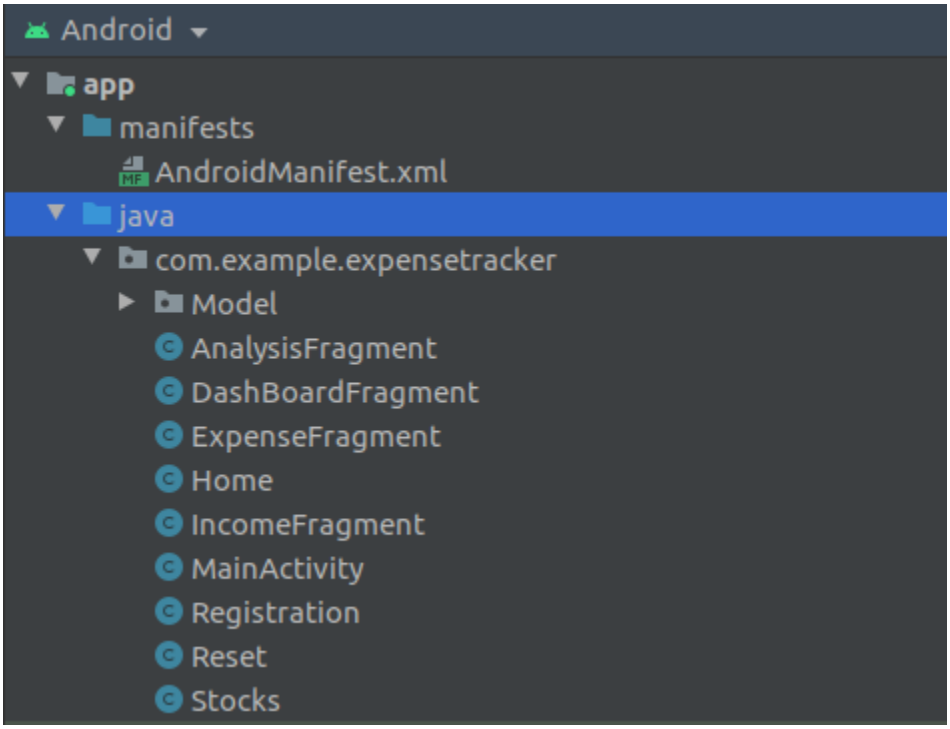

Figure 41 List of activities in our project

We have stored all these logs in the separate log file. You can find the logfile in github repositoy. Direct link for same is:
https://github.com/printSamarth/ExpenseTracker/blob/master/ExpenseTracker.log

ExpenseTracker.log

```
05-03 09:04:58.763 11419 11419 I Registration: Registration..
05-03 09:04:58.763 11419 11419 I Registration: Username collected.
05-03 09:04:58.763 11419 11419 I Registration: Password collected.
05-03 09:04:58.764 11419 11419 I Registration: Processing..
05-03 09:04:58.921 11419 11419 E Registration: Registration Failed username already exists
05-03 09:05:01.752 11419 11419 I Registration: Move to sign in page
05-03 09:07:16.817 11903 11903 I MainActivity: Inside on create Method
05-03 09:07:17.192 11903 11903 I MainActivity: Inside on create Method
05-03 09:07:27.436 11903 11903 I MainActivity: Username collected
05-03 09:07:27.436 11903 11903 I MainActivity: Password collected
05-03 09:07:27.436 11903 11903 I MainActivity: Validating username and password
05-03 09:42:03.070 12265 12265 I IncomeFragment: Inside on create method
05-03 09:42:03.124 12265 12265 I IncomeFragment: Calcuating total income of current user
05-03 09:42:03.126 12265 12265 I IncomeFragment: Total Income - 12400
05-03 09:46:22.491 12265 12265 I IncomeFragment: Inside on create method
05-03 09:46:22.534 12265 12265 I IncomeFragment: Calcuating total income of current user
05-03 09:46:22.536 12265 12265 I IncomeFragment: Total Income - 12400
05-03 09:42:32.826 12265 12265 I ExpenseFragment: Inside on create method
05-03 09:42:32.849 12265 12265 I ExpenseFragment: Calcuating total Expense of current user
05-03 09:42:32.850 12265 12265 I ExpenseFragment: Total Expense - 4000
05-03 09:46:24.893 12265 12265 I ExpenseFragment: Inside on create method
05-03 09:46:24.912 12265 12265 I ExpenseFragment: Calcuating total Expense of current user
05-03 09:46:24.914 12265 12265 I ExpenseFragment: Total Expense - 4000
05-03 09:43:08.399 12265 12265 I AnalysisFragment: Inside oncreate method
05-03 09:43:08.425 12265 12265 I AnalysisFragment: calling load data function
05-03 09:43:08.425 12265 12265 I AnalysisFragment: Inside load data function
05-03 09:43:08.425 12265 12265 I AnalysisFragment: Maximum spending is done on - Clothes
05-03 09:39:33.657 12265 12265 I DashBoardFragment: mExpense 0
05-03 09:39:33.657 12265 12265 I DashBoardFragment: Bank Balance 0.0
05-03 09:39:33.658 12265 12265 I DashBoardFragment: Cash Left 0.0
05-03 09:39:33.713 12265 12265 I DashBoardFragment: mIncome 1
05-03 09:39:33.713 12265 12265 I DashBoardFragment: Bank Balance  0.0
05-03 09:39:33.713 12265 12265 I DashBoardFragment: Cash Left 0.0
05-03 09:39:39.617 12265 12265 I DashBoardFragment: Inside Income data insert function
05-03 09:39:40.076 12265 12265 I DashBoardFragment: Payment Medium - Cash
05-03 09:39:40.076 12265 12265 I DashBoardFragment: type - Salary
05-03 09:39:46.166 12265 12265 I DashBoardFragment: Payment Medium - Bank
05-03 09:39:54.382 12265 12265 I DashBoardFragment: Inserting Expense Transaction data
```

Figure 42 Logfile

Now, we need to prepare one configuration file which tells logstash to find this logfile and format of log statements. So, logstash.conf will contain the path to the logfile and format of log statements. Logstash configuration file for our project is as shown below:

```
input{
        file{
                path => "/home/samarth/ExpenseTracker/ExpenseTracker.log"
                start_position => "beginning"
                sincedb_path => "/dev/null"
                type => "log"
        }
}
filter {
 dissect {
   mapping => {
     "message" => "%{date} %{time} %{process_id} %{thread_id} %{package_priority} %{tag} %{tag_message}"
    }
   remove_field => ["message"]
  }
}
output {
        elasticsearch{hosts => ["localhost:9200"] index=> "app_elastic" }
        stdout {codec => rubydebug}
}
```

Figure 43 Logstash configuration file

Now, we are ready to analyze these logs using ELK stack. Logstash uses this logfile to generate anayasis charts as per required. Kibana generates charts. For Kibana we need elastic search server.

Elasticsearch runs on port number 9200, Kibana runs on port number 5601 and logstash runs on port number 9600. Setting up kibana includes creating a new index pattern under management. Add a new index pattern and press next to choose @timestamp from next window and this will create a new index pattern to view your logs.

Follow below commands to run elastic search, kibana and logstash respectively:

$./path_to_elasticsearch/bin/elasticsearch

```
samarth@samarth-Inspiron-5447:~/Downloads/elasticsearch-7.12.1$ ./bin/elasticsearch
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
Future versions of Elasticsearch will require Java 11; your Java version from [/usr/lib/jvm/java-8-openjdk-amd64/jre] does not meet this requirement. Consider switching to a distribution of Elasticsearch with a bundled JDK. If you are already using a distribution with a bundled JDK, ensure the JAVA_HOME environment variable is not set.
warning: usage of JAVA_HOME is deprecated, use ES_JAVA_HOME
Future versions of Elasticsearch will require Java 11; your Java version from [/usr/lib/jvm/java-8-openjdk-amd64/jre] does not meet this requirement. Consider switching to a distribution of Elasticsearch with a bundled JDK. If you are already using a distribution with a bundled JDK, ensure the JAVA_HOME environment variable is not set.
[2021-05-13T11:48:38,571][INFO ][o.e.n.Node               ] [samarth-Inspiron-5447] version[7.12.1], pid[3869], build[default/tar/3186837139b9c6b6d23c3200870651f10d3343b7/2021-04-20T20:56:39.040728659Z], OS[Linux/5.8.0-50-generic/amd64], JVM[Private Build/OpenJDK 64-Bit Server VM/1.8.0_292/25.292-b10]
```

Figure 44 run elasticsearch server

$./path_to_kibana/bin/kibana

```
samarth@samarth-Inspiron-5447:~/Downloads/kibana-7.11.1-linux-x86_64$ ./bin/kibana
  log   [11:50:44.931] [info][plugins-service] Plugin "visTypeXy" is disabled.
  log   [11:50:45.107] [warning][config][deprecation] Config key [monitoring.cluster_alerts.email_n
otifications.email_address] will be required for email notifications to work in 8.0."
  log   [11:50:45.518] [info][plugins-system] Setting up [101] plugins: [taskManager,licensing,glob
alSearch,globalSearchProviders,code,usageCollection,xpackLegacy,telemetryCollectionManager,telemetr
y,telemetryCollectionXpack,kibanaUsageCollection,securityOss,newsfeed,mapsLegacy,kibanaLegacy,trans
lations,share,legacyExport,embeddable,uiActionsEnhanced,expressions,charts,esUiShared,bfetch,data,h
ome,observability,console,consoleExtensions,apmOss,searchprofiler,painlessLab,grokdebugger,manageme
nt,indexPatternManagement,advancedSettings,fileUpload,savedObjects,visualizations,visTypeVislib,vis
TypeVega,visTypeTimelion,features,licenseManagement,dataEnhanced,watcher,canvas,visTypeTimeseries,v
isTypeTimeseriesEnhanced,regionMap,visTypeTagcloud,visTypeTable,visTypeMetric,visTypeMarkdown,tileM
ap,mapsOss,lensOss,inputControlVis,graph,timelion,dashboard,dashboardEnhanced,visualize,discover,di
scoverEnhanced,savedObjectsManagement,spaces,security,reporting,savedObjectsTagging,maps,lens,lists
,encryptedSavedObjects,dashboardMode,cloud,upgradeAssistant,snapshotRestore,fleet,indexManagement,r
```

Figure 45 Kibana running

$./path_to_logstash/bin/logstash -f path_to_configfile/logstash.conf

```
{
                "date" => "05-03",
                "path" => "/home/samarth/ExpenseTracker/ExpenseTracker.log",
           "thread_id" => "11419",
                "time" => "09:04:58.763",
    "package_priority" => "I",
                 "tag" => "Registration:",
                "host" => "samarth-Inspiron-5447",
          "@timestamp" => 2021-05-13T06:25:49.148Z,
                "type" => "log",
            "@version" => "1",
          "process_id" => "11419",
         "tag_message" => "Registration.."
}
{
                "date" => "05-03",
                "path" => "/home/samarth/ExpenseTracker/ExpenseTracker.log",
           "thread_id" => "11419",
                "time" => "09:04:58.763",
    "package_priority" => "I",
                 "tag" => "Registration:",
                "host" => "samarth-Inspiron-5447",
          "@timestamp" => 2021-05-13T06:25:49.230Z,
                "type" => "log",
            "@version" => "1",
          "process_id" => "11419",
         "tag_message" => "Username collected."
}
{
                "date" => "05-03",
                "path" => "/home/samarth/ExpenseTracker/ExpenseTracker.log",
           "thread_id" => "11419",
                "time" => "09:05:01.752",
    "package_priority" => "I",
                 "tag" => "Registration:",
                "host" => "samarth-Inspiron-5447",
          "@timestamp" => 2021-05-13T06:25:49.235Z,
                "type" => "log",
```

Figure 46 Logstash running

Now, in the browser open localhost on port number 5601. You can see Kibana dashboard. In the menu go to stack management under management. The reference image is given below:

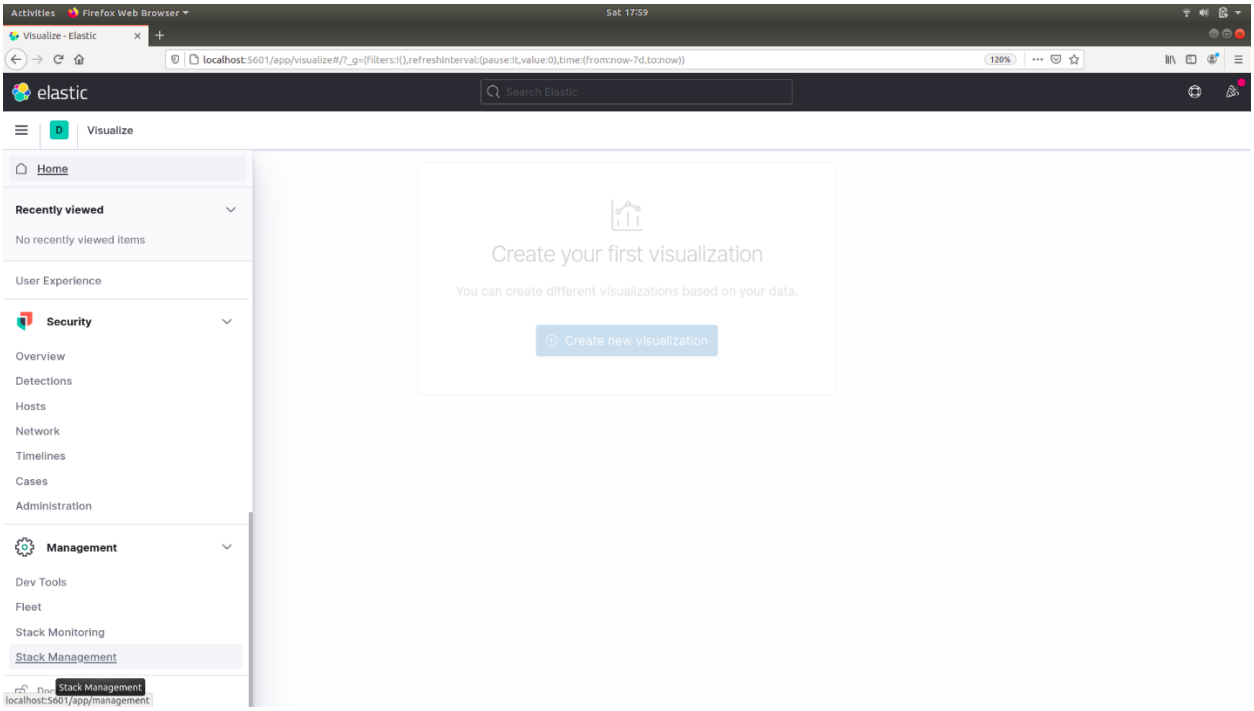

Figure 47 Stack management

Under stack management go to index pattern under Kibana section as shown below:

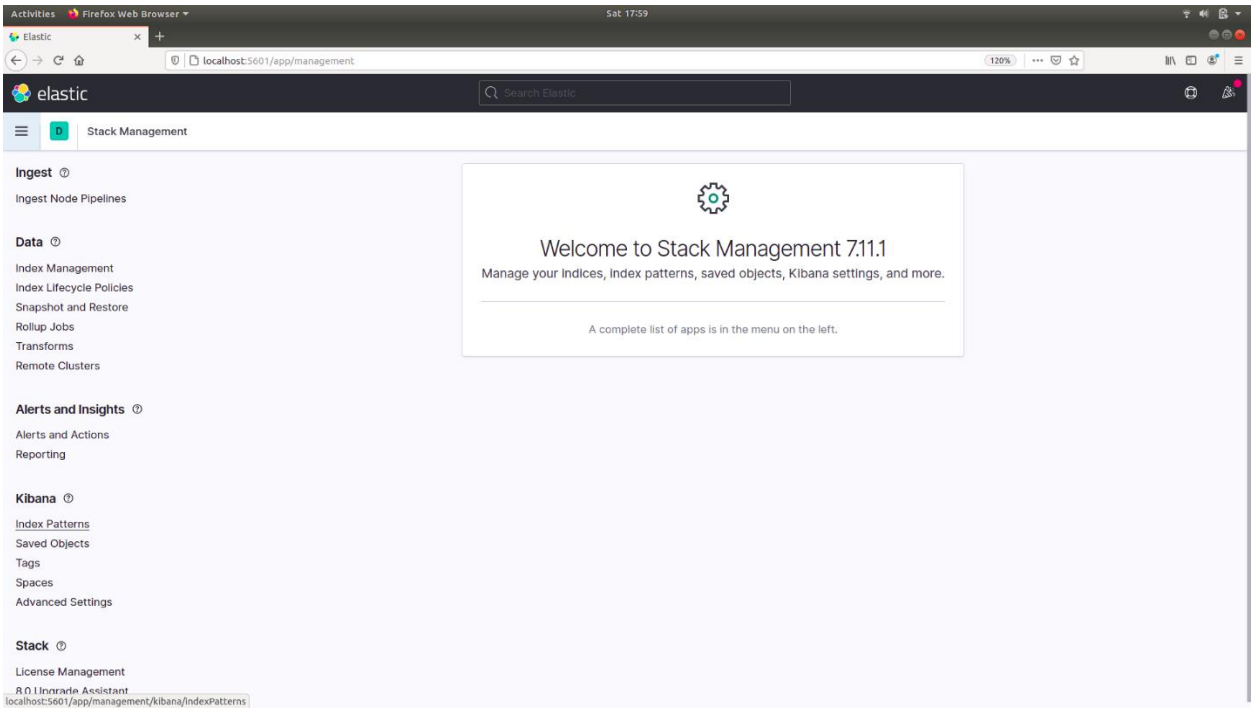

Figure 48 Select index pattern

Now, create index pattern for timestamp as shown below:

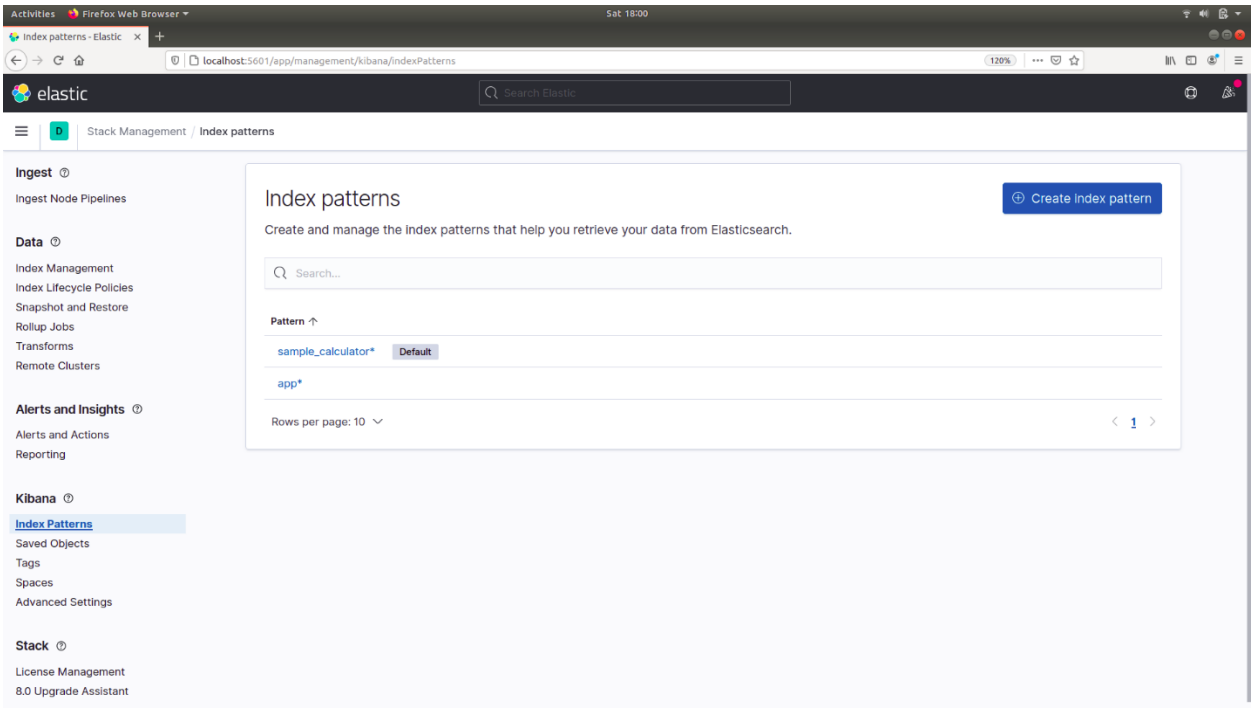

Figure 49 Specify index pattern

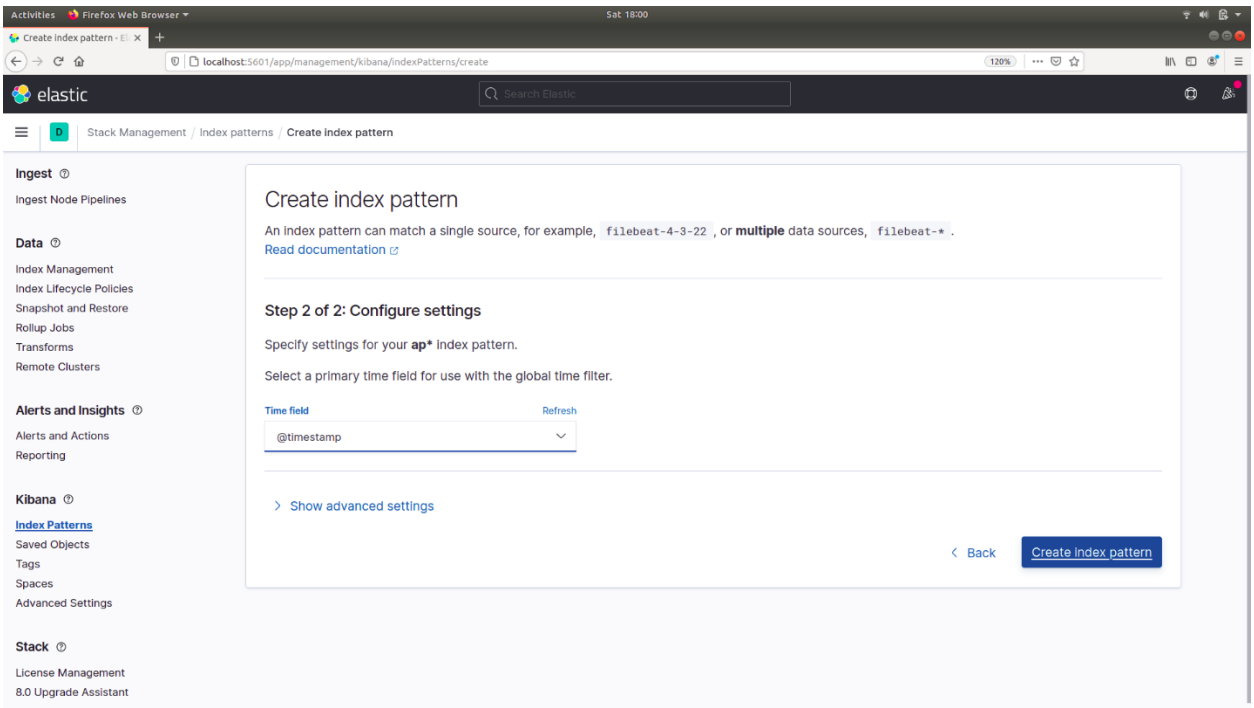

Figure 50 Create index pattern

You can find timestamp graph for your application under discover section:

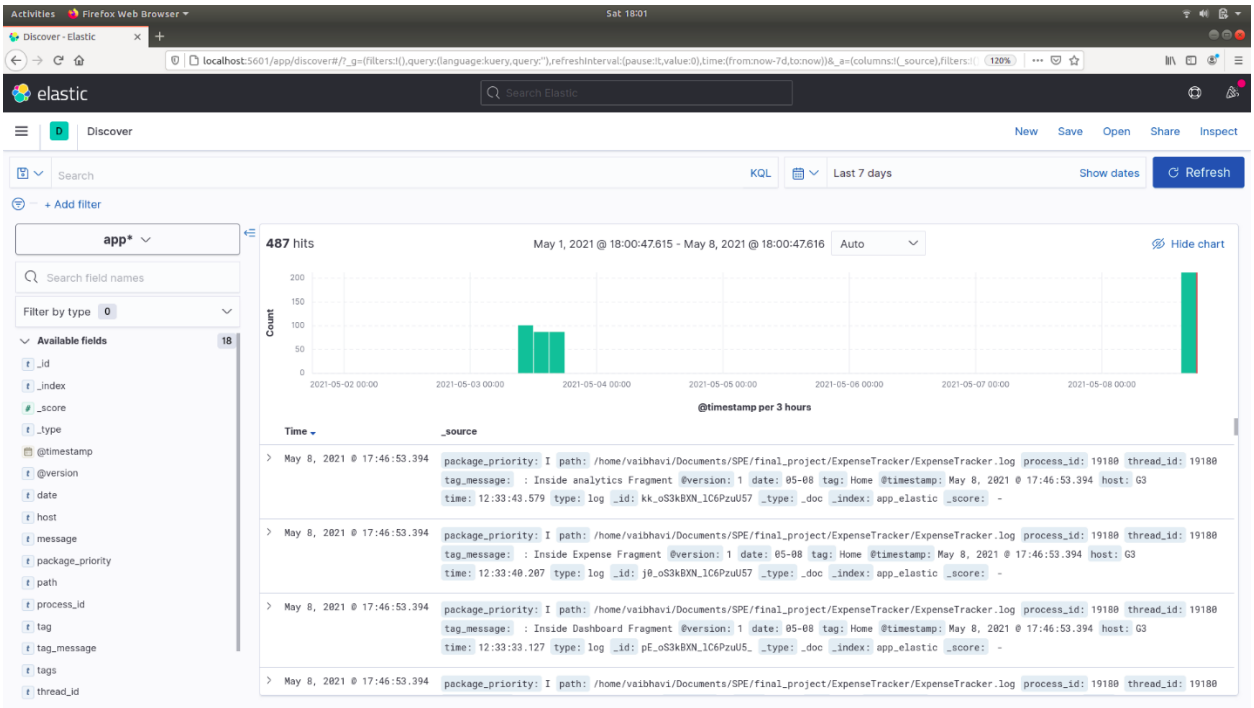

Figure 51 Discover timestamp

To create charts based on your log statements, go to visualize section and choose lens:

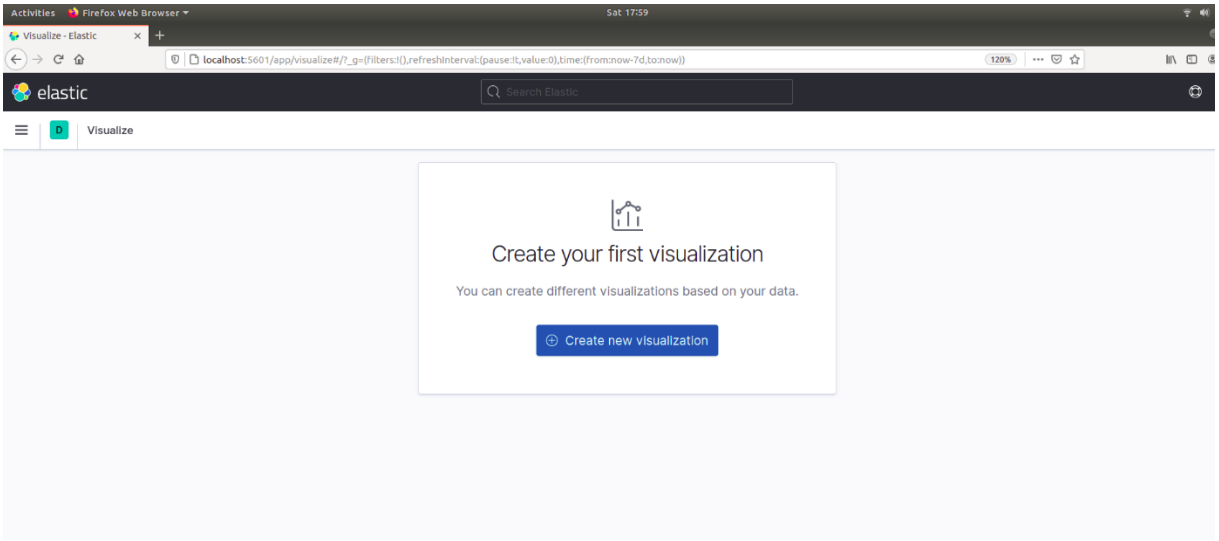

Figure 52 Create new visualization

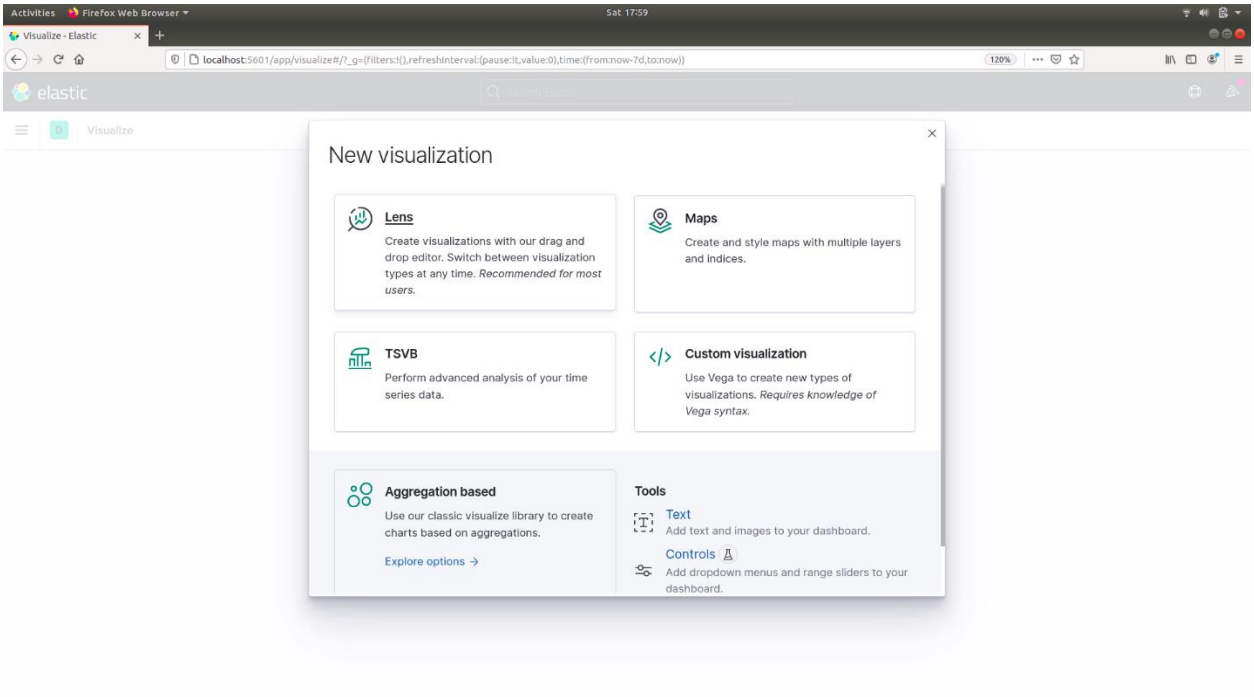

Figure 53 Select Lens

Now, specify the parameters and charts to visualize the information. Different charts for our application are shown below:

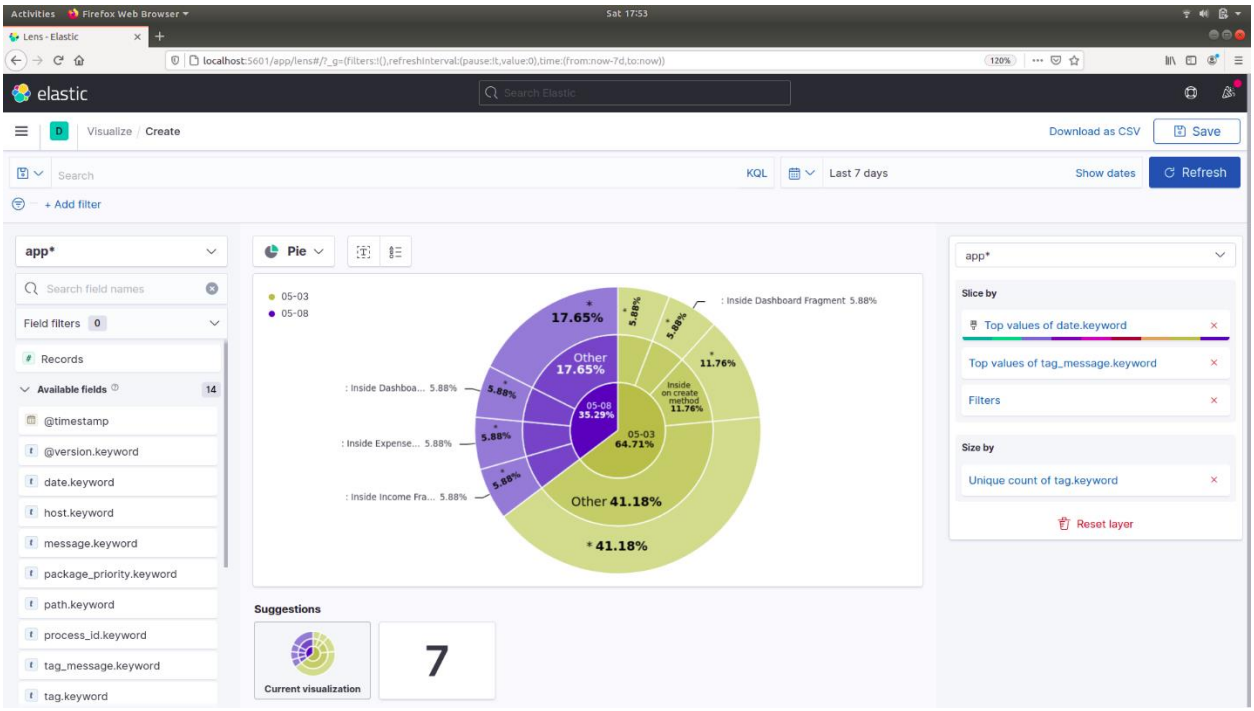

Figure 54 Analysis

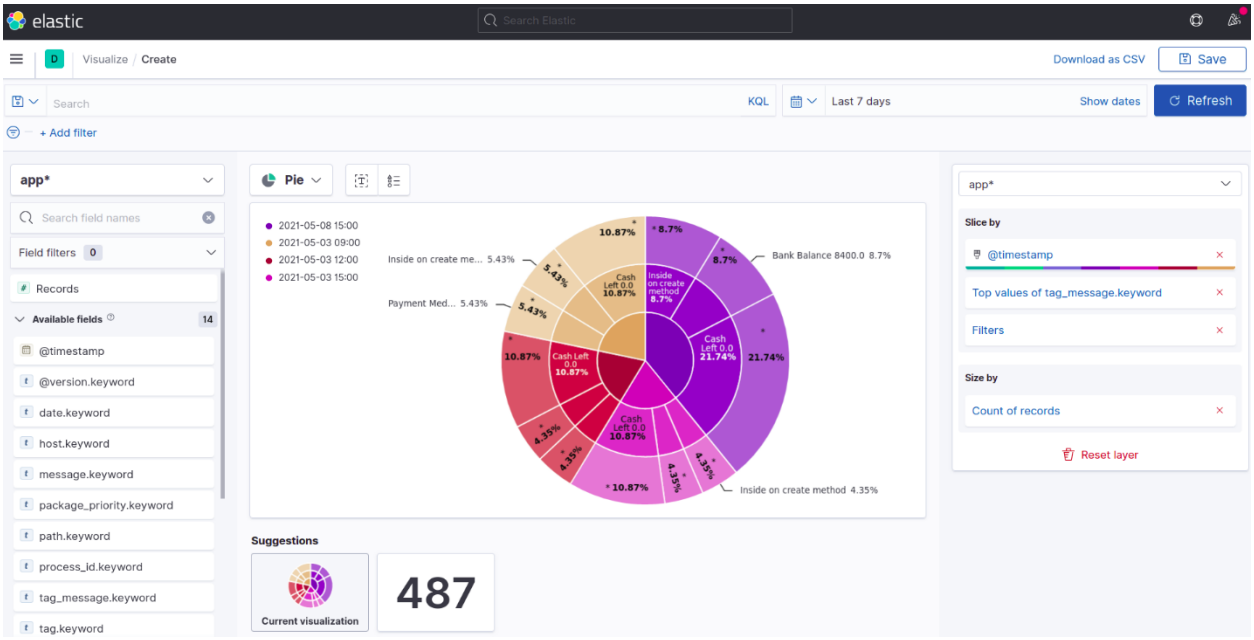

Figure 55 anaysis

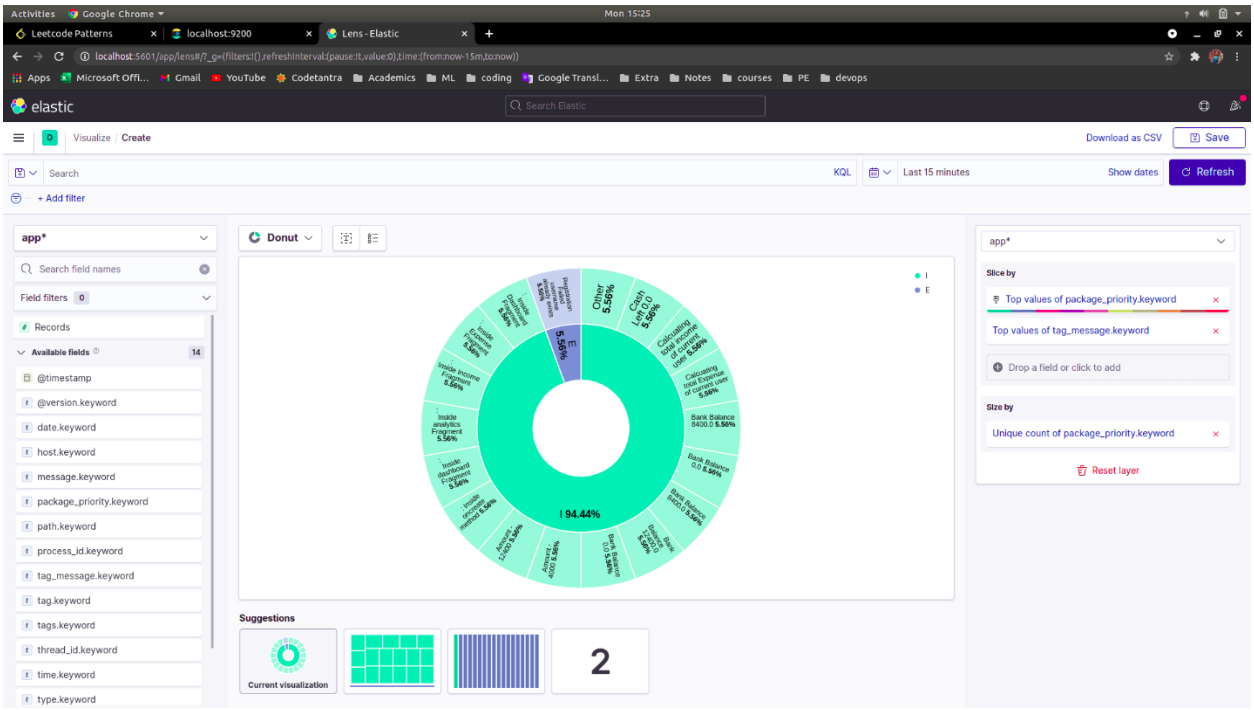

Figure 56 Analysis

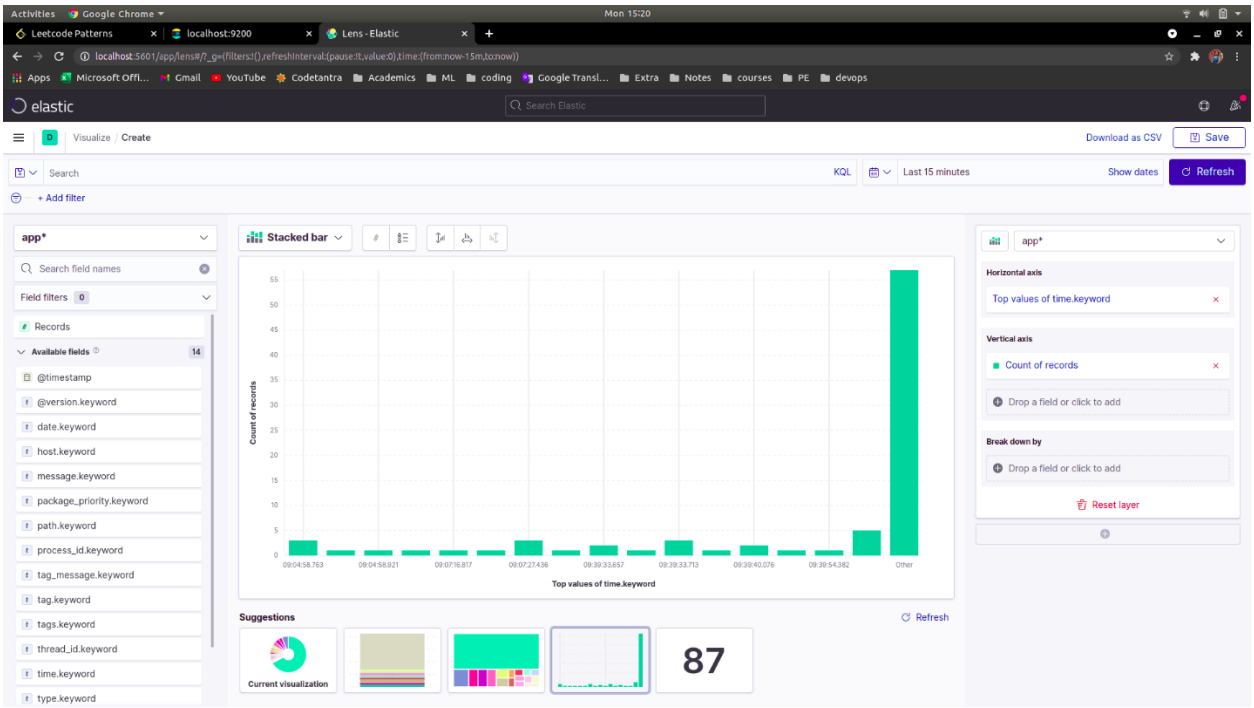

Figure 57 Analysis

4.8 Building pipeline and running on deployed machine

Pipeline script

Script

```
pipeline {
    environment{
        imageName = ''
    }
    agent any

    stages {
        stage('Git'){
            steps{
                git 'https://github.com/printSamarth/ExpenseTracker'

            }
        }
        stage('Clean') {
            steps {
                sh './gradlew --refresh-dependencies clean'
            }
        }
        stage('Build'){
```

Figure 58 Final Jenkins Pipeline

Pipeline script

Script

```
19        stage('Build'){
20            steps{
21                sh './gradlew assembleDebug'
22                archiveArtifacts '**/app-debug.apk'
23            }
24        }
25        stage('Install for testing'){
26            steps{
27                sh './gradlew installDebug'
28            }
29        }
30        stage('Testing'){
31            steps{
32                sh './gradlew connectedDebugAndroidTest'
33            }
34        }
35        stage('Docker build to Image') {
36                steps {
37                    script{
38                        imageName = docker.build "ssvapp/expense-tracker:latest"
```

Figure 59 Final Jenkins Pipeline

Pipeline script

Script

```
35        stage('Docker build to Image') {
36                steps {
37                    script{
38                        imageName = docker.build "ssvapp/expense-tracker:latest"
39                    }
40                }
41            }
42            stage('Push Docker Image to DockerHub') {
43                steps {
44                    script{
45                        docker.withRegistry('','7fd7ec98-6ea4-4c67-9d94-a9cfdb4592c6')
46                        {
47                            imageName.push()
48                        }
49                    }
50                }
51            }
52            stage('Pull Docker Image') {
53                steps {
54
```

Figure 60 Final Jenkins Pipeline

Pipeline script

Script

```
49                    }
50                }
51            }
52            stage('Pull Docker Image') {
53                steps {
54                    ansiblePlaybook becomeUser: null, colorized: true, disableHostKeyChecking: true, installation: 'Ansible',
55                    inventory: 'deploy-docker/inventory', playbook: 'deploy-docker/deploy-image.yml',
56                    sudoUser: null
57                }
58            }
59        stage('DISTRIBUTE') {
60            steps {
61                appCenter apiToken: '92a01f4cb9c3bb7a57de2984bde44c67b00f5979',
62                        ownerName: 'ssvapp',
63                        appName: 'ExpenseTracker',
64                        pathToApp: '**/app-debug.apk',
65                        distributionGroups: 'tester'
66            }
67        }
68    }
69 }
```

Figure 61 Final Jenkins Pipeline

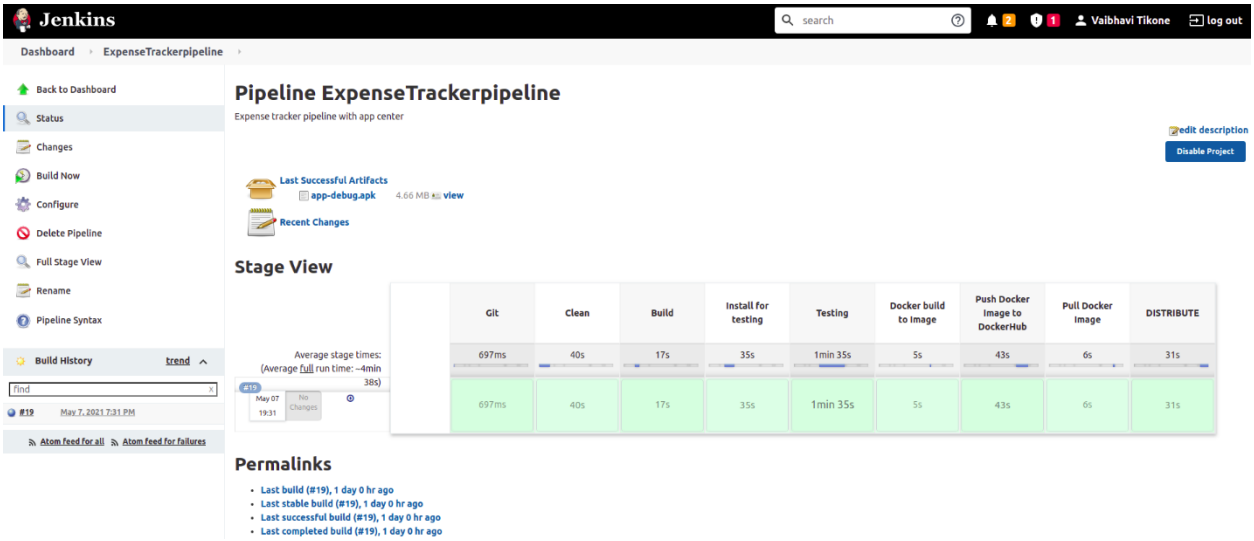

Figure 62 Final Jenkins pipeline build states

5. Experimental setup

5.1 App features

Expense tracker app has following features:

- User sign in and sign up.
- Adding expense data: User can add expense data with certain parameters like mode of payment, spent on category, description.
- Adding income data: User can add income data with certain parameters like mode of payment, type of income.
- Updating income and expense data: All the expense data will get listed and user can do update operation by clicking on it.
- Tracking total income, total expenditure, net bank balance and net cash balance.
- Analysis based on expense data: User can see their spending per category and apply date filter. Moreover, they can see comparison with previous spending.
- Tracking stocks: User can add his stocks names with quantity and can see real-time valuation of his portfolio.

5.2 Architecture diagram

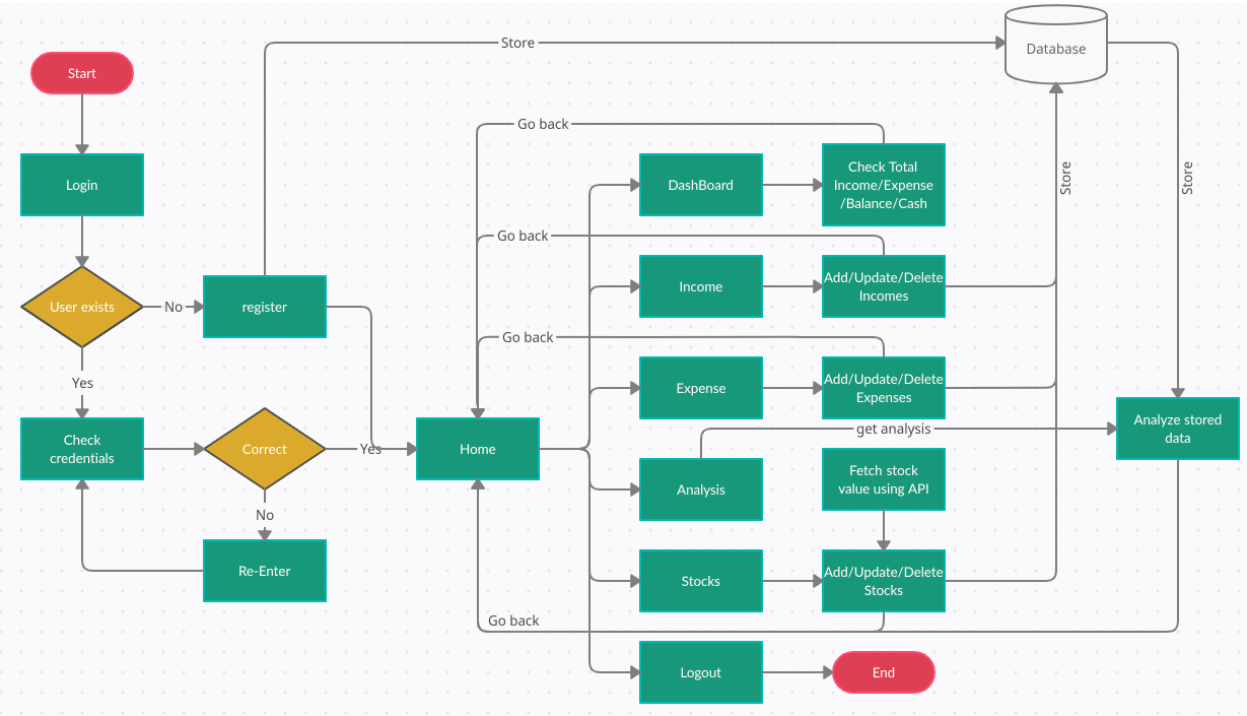

Figure 63 Architecture diagram

6. Result and Discussion

6.1 Login page

On starting the app, log in page will be seen first. If the user does not have an account, he can create it by clicking on “Don’t have an account ?” button.

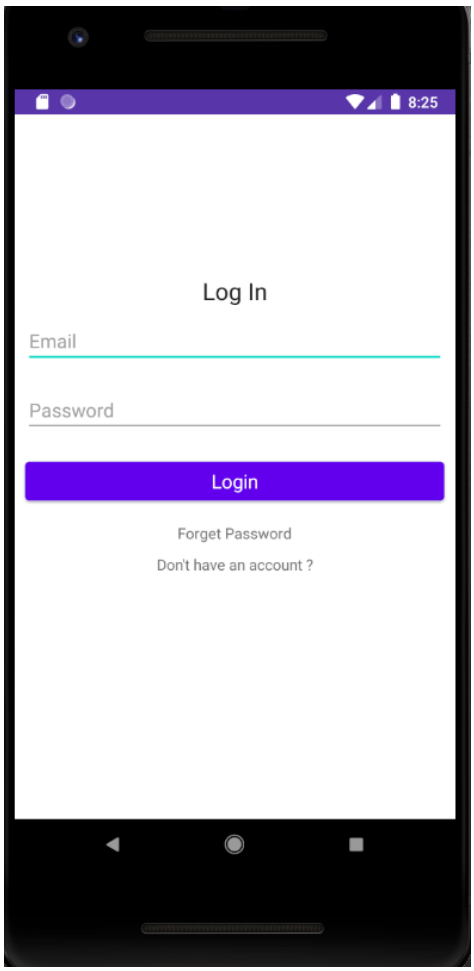

Figure 64 Login page

6.2 Dashboard page

This page shows the total income and total spending user has done so far. Along with it, user's net from bank and cash has also been shown here.

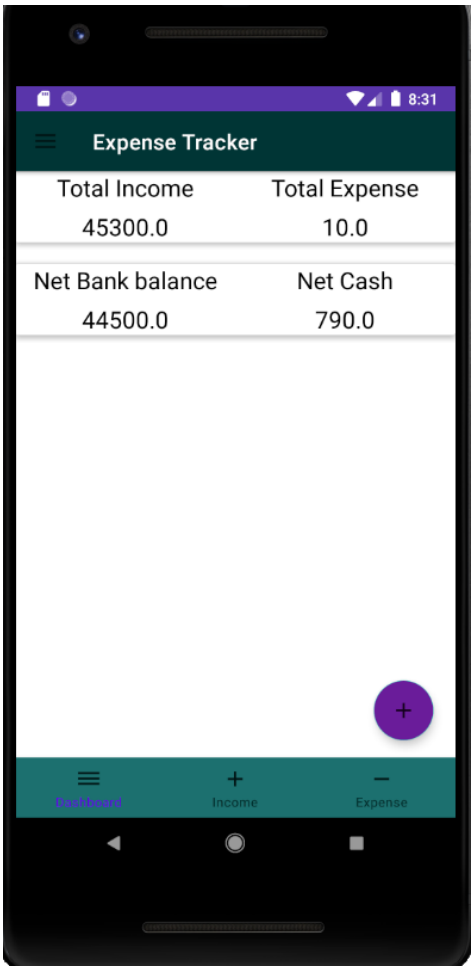

Figure 65 Dashboard page

6.3 Menu

On clicking at the menu icon on top left corner, user will get a sliding menu from which user can navigate easily.

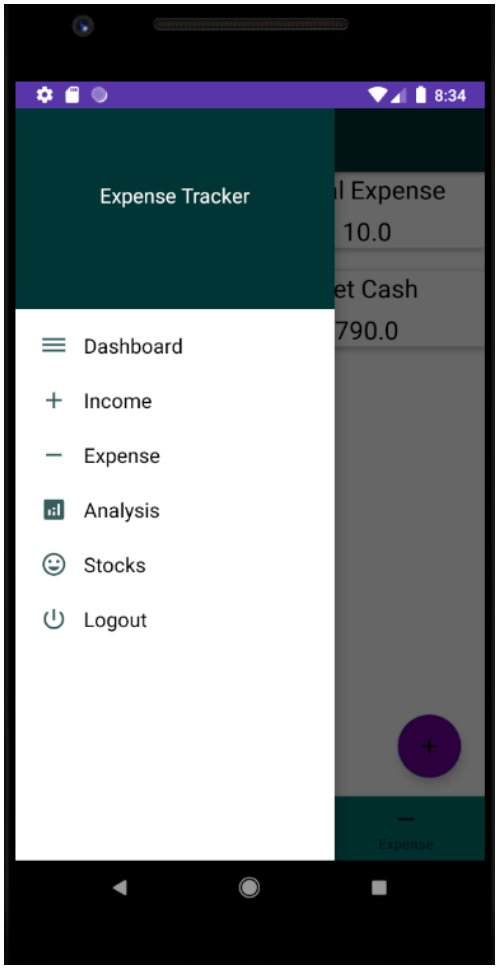

Figure 66 Menu

6.4 Adding income and expense data

From dashboard, by clicking on bottom floating button user can add income and expense data.

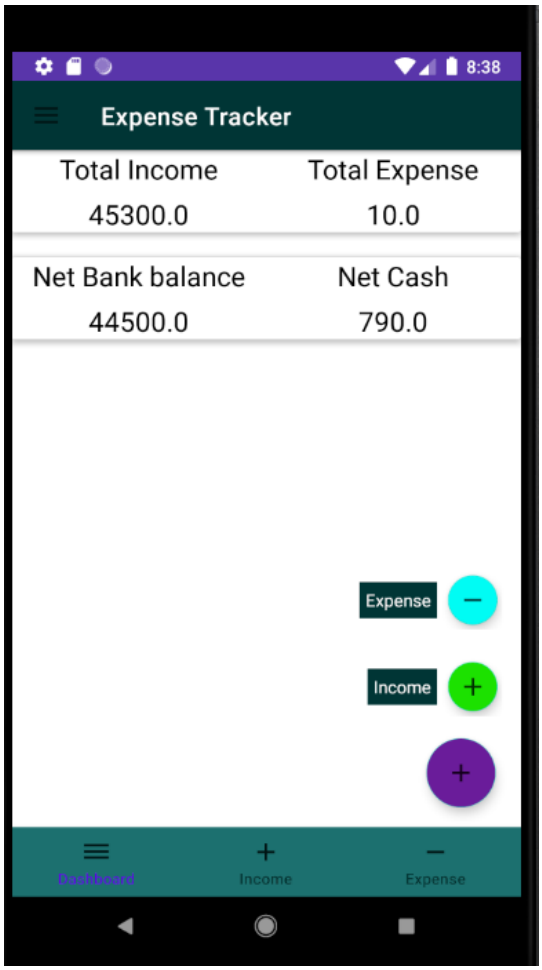

Figure 67 Adding Income/Expense

Now on clicking plus icon, user gets a floating dialogue box from which he can add income related information.

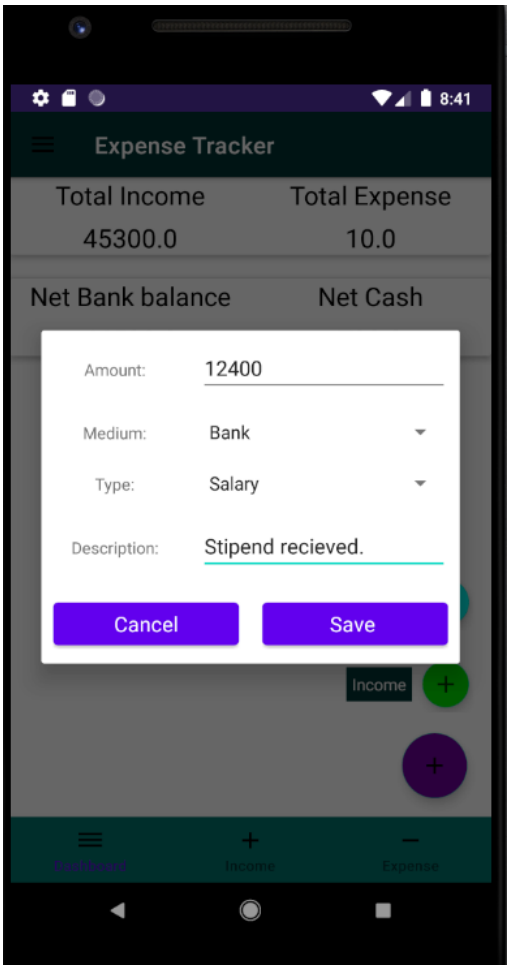

Figure 68 adding Income details

And on clicking minus icon, user gets a floating dialogue box from which he can add expense related information.

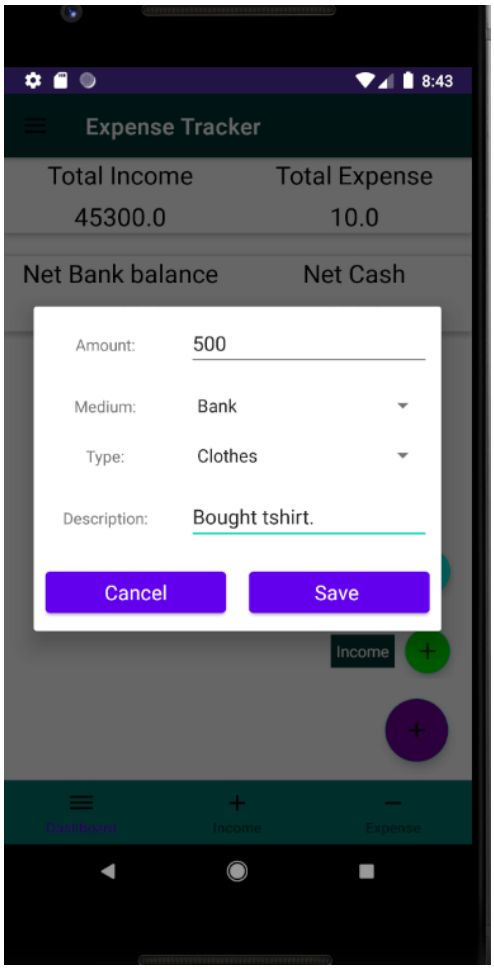

Figure 69 Adding Expense details

6.5 Income page

Here user can see his past income related information and also do update and delete operations.

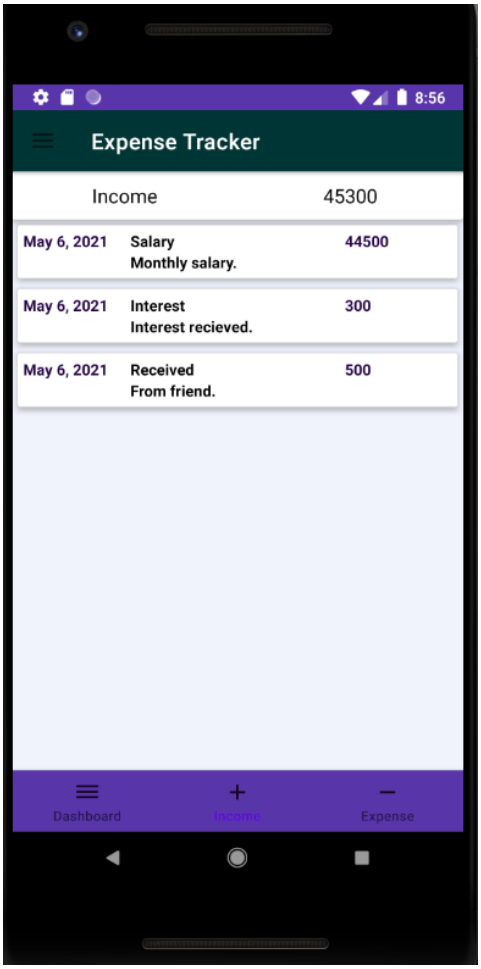

Figure 70 Income page

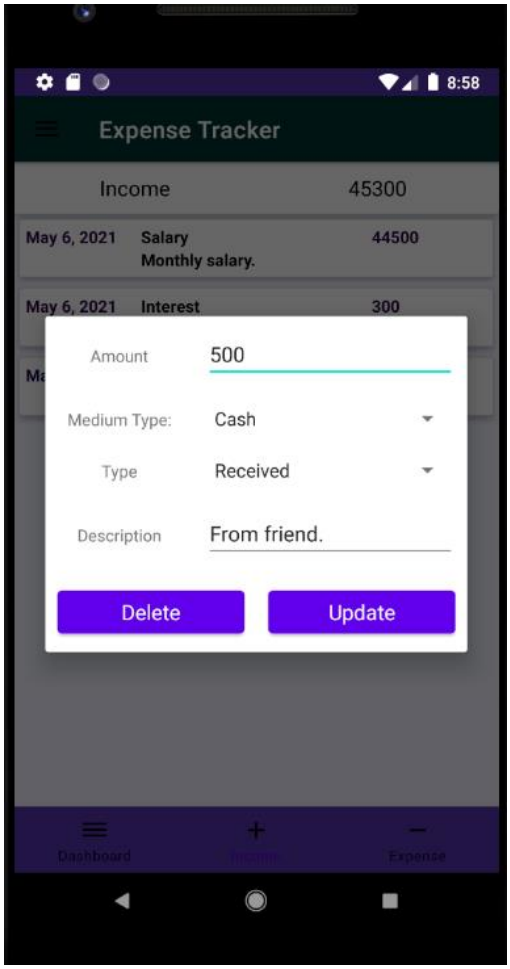

Figure 71 Updating Income details

6.6 Expense page

Here user can see past expense related information and can do update and delete operation on past data.

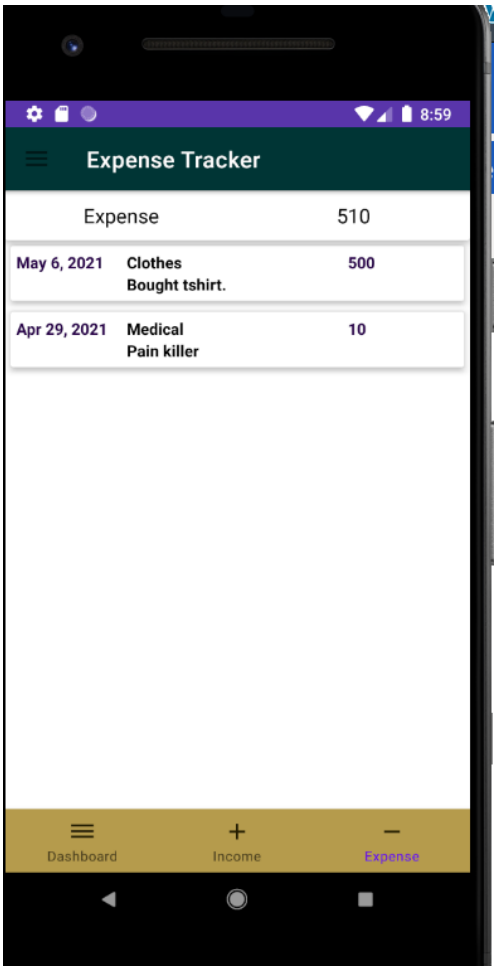

Figure 72 Expense page

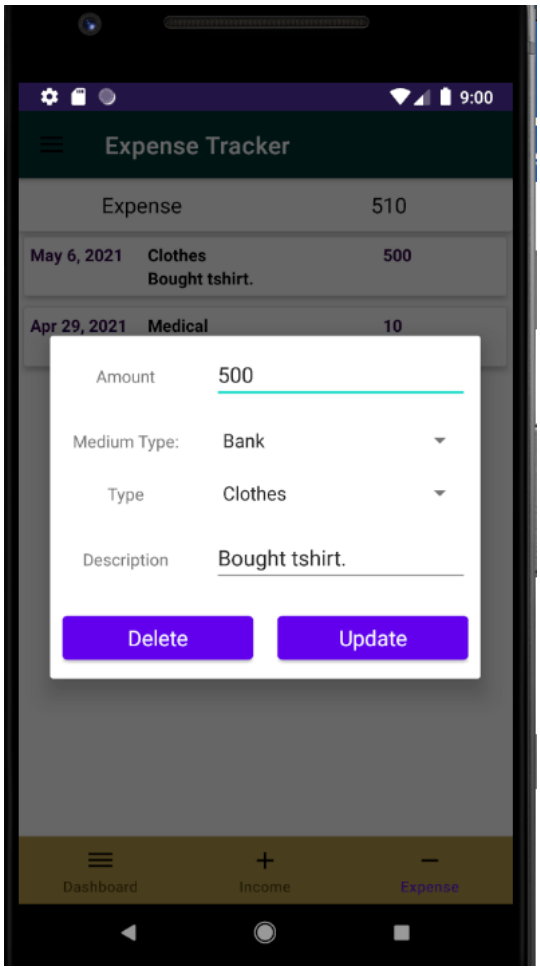

Figure 73 Updating Expense details

6.7 Analysis page

User can navigate to analysis page and there he can see visualization of his past expense. And here he can also apply date filter to see analysis over certain period.

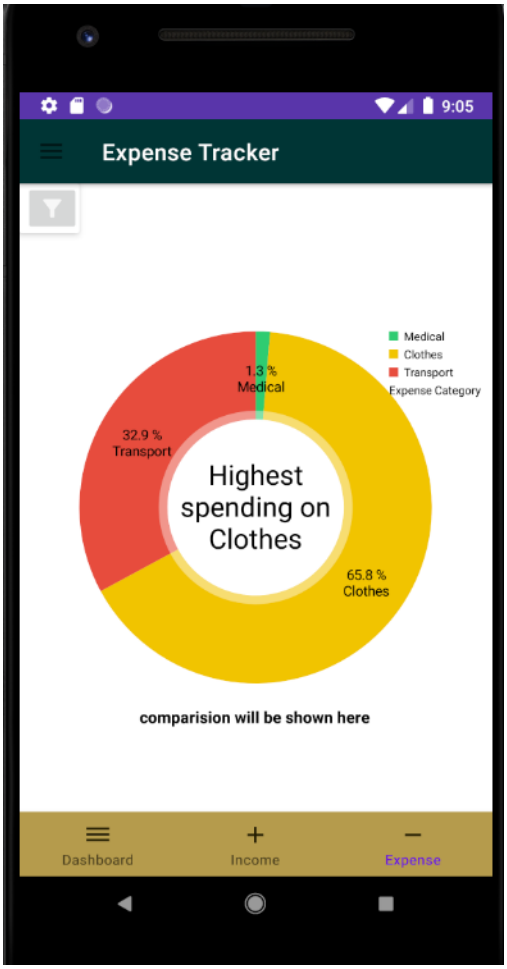

Figure 74 Analysis Page

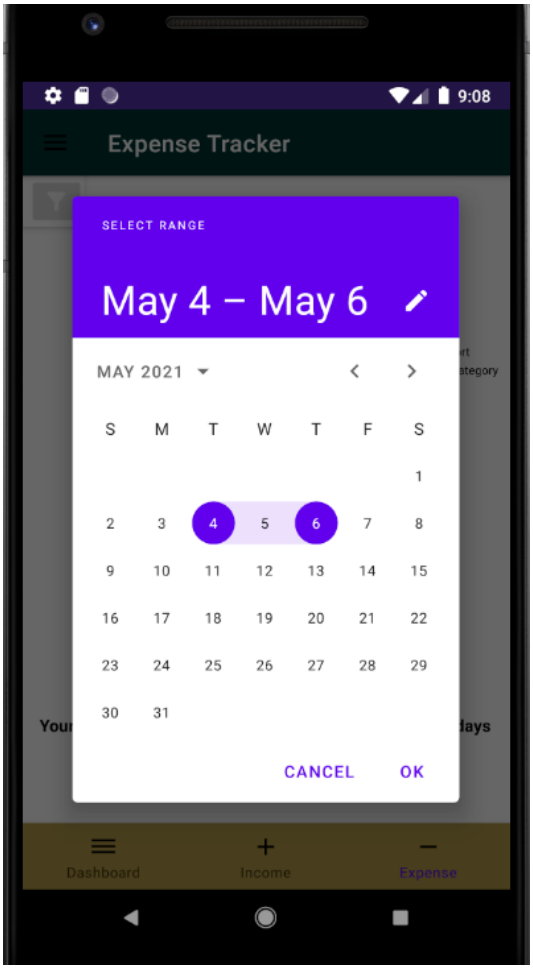

Figure 75 Generating custom analysis

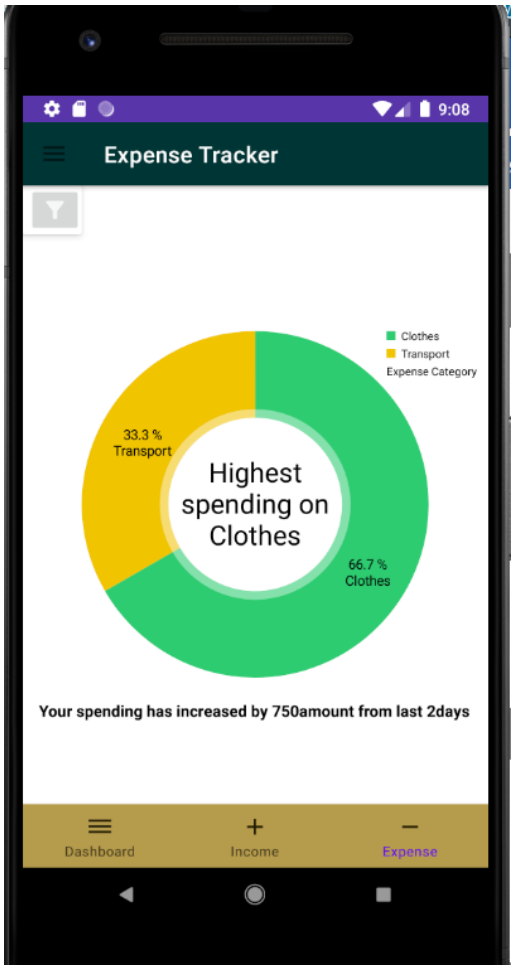

Figure 76 Custom analysis

6.8 Stocks page

Here user can see total valuation of each stock based on current price and number of units. Also, he can add new stocks and do changes in already present stocks.

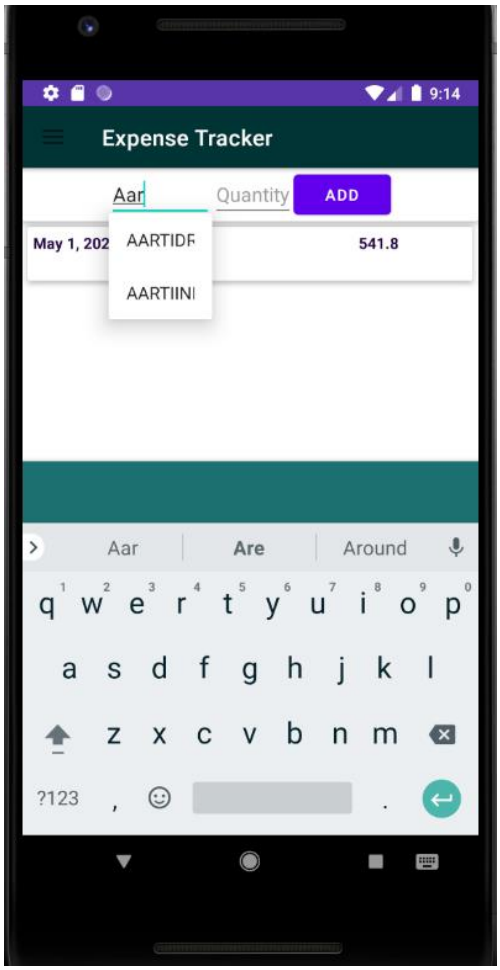

Figure 77 Searching stocks

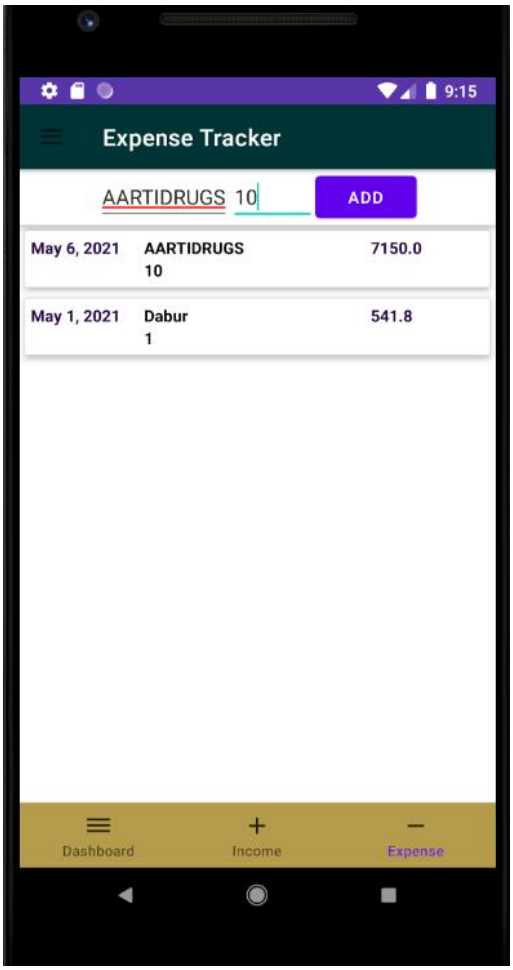

Figure 78 Adding stocks

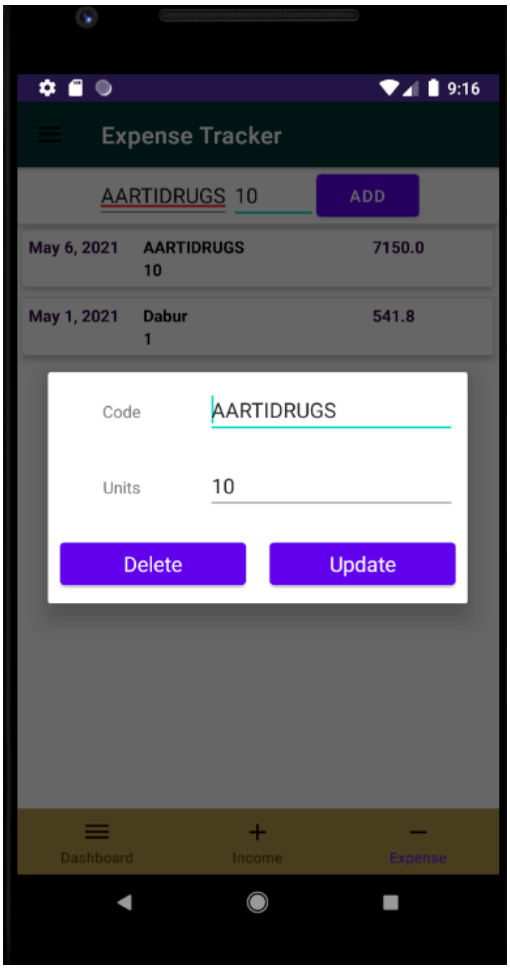

Figure 79 Updating stocks

6.9 Logout

User can click on logout button from menu to logout of the system.

7. Scope for future work

7.1 OCR to scan bills

Rather than taking just text data as input, we could take images of bill as an input and that image we will send to python server where we will run OCR model to extract all information and python server will return json object which we will store in our database. As an experiment we had tried using tesseract to do OCR, but as it is not able to do OCR on tabular and handwritten data.

7.2 Stock portfolio management

Based on current stocks user has and amount invested in it, we could give user suggestions to buy new stocks in a way that it makes users portfolio diversified.

8. Concusion

We have successfully built an android application for managing expenses with analysis along with stocks valuation using devops tools. Devops tools we used are: Git, Gradle, Jenkins, Espresso, Docker, Ansible, and app center. Devops tools helped to build CI/CD pipeline successfully.
The Devops methodology and lifecycle tools prove to be better than the Agile methodology in terms of technical, cultural and business benefits. By minimizing friction between independent teams, DevOps enables a collaborative approach for enterprise software development and delivery that reflects the needs of the entire application lifecycle for today's modern enterprises. Thus, we can develop, test, deploy and monitor the application easily.

9. References

[1] Install Android stuido: https://developer.android.com/studio/install
[2] Install git on windows: https://phoenixnap.com/kb/how-to-install-git-windows
[3] Install docker: https://docs.docker.com/engine/install/
[4] Jenkins setup for android: https://www.youtube.com/watch?v=oi67zoyx5fY
[5] CII/CD pipeline for android app using Jenkins: https://www.youtube.com/watch?v=ONmjFu3-3go
[6] Python script in docker: https://stackabuse.com/dockerizing-python-applications/