

**INTERNATIONAL INSTITUTE OF INFORMATION  
TECHNOLOGY-BANGALORE**



**CS 816 - SOFTWARE PRODUCTION ENGINEERING**

**Final Project**

---

# **Expense Sharing WebApp**

**Team Members:-**

Aman Gupta (MT2021012)

Kunal Sharma (MT2021070)

Nikhil Mittal (MT2021083)

**Under Guidance of:-**

Prof. B. Thangaraju

**Teaching Assistant:-**

Anshul Garg



# Table of Contents

|  |          |
|--|----------|
| <b>Table of Contents</b>                 | <b>2</b> |
| <b>1. Abstract</b>                       | <b>4</b> |
| <b>2. Introduction</b>                   | <b>5</b> |
| 2.1. Overview                            | 5        |
| 2.2. Features                            | 5        |
| 2.3. Why DevOPs                          | 6        |
| <b>3. System Configuration</b>           | <b>7</b> |
| 3.1. Operating System                    | 7        |
| 3.2. CPU and RAM                         | 7        |
| 3.3. Language                            | 7        |
| 3.4. Kernel Machine                      | 7        |
| 3.5. Database                            | 7        |
| 3.6. Building Tools                      | 7        |
| 3.7. DevOps Tools                        | 7        |
| <b>4. Software Development Lifecycle</b> | <b>8</b> |
| 4.1. Installations                       | 8        |
| 4.1.1. SpringBoot                        | 8        |
| 4.1.2. Angular                           | 9        |
| 4.1.3. MySQL                             | 10       |
| 4.1.4. Docker                            | 10       |
| 4.2. Source Control Management           | 11       |
| 4.2.1. CI Jenkins Pipeline               | 13       |
| 4.3. Build                               | 14       |
| 4.3.1. Maven                             | 14       |
| 4.3.2. CI Pipeline                       | 15       |
| 4.4. Testing                             | 15       |

|  |           |
|--|-----------|
| 4.5. Docker Artifact                                   | 16        |
| 4.5.1. CI Pipeline                                     | 18        |
| 4.5.2. Docker Compose                                  | 21        |
| 4.6. Continuous Deployment                             | 22        |
| 4.6.1. CI Pipeline                                     | 23        |
| 4.7. Monitor   | 24        |
| 4.8. Building Pipeline and Running on Deployed Machine | 27        |
| <br>   |           |
| <b>5. Experimental Setup</b>                           | <b>28</b> |
| 5.1. Functional Requirements                           | 28        |
| 5.2. Class Diagram                                     | 29        |
| 5.3. API Documentation                                 | 30        |
| 5.4. Code Walkthrough                                  | 31        |
| <br>   |           |
| <b>6. Results and Discussion</b>                       | <b>35</b> |
| 6.1. Home Page   | 35        |
| 6.2. Sign Up   | 35        |
| 6.3. Sign In   | 36        |
| 6.4. User Dashboard                                    | 36        |
| 6.5. Group Details                                     | 37        |
| 6.6. Adding Expenses                                   | 37        |
| 6.7. Adding Group                                      | 38        |
| <br>   |           |
| <b>7. Scope of Future Work</b>                         | <b>38</b> |
| <br>   |           |
| <b>8. Conclusion</b>                                   | <b>39</b> |
| <br>   |           |
| <b>9. References</b>                                   | <b>40</b> |

---

# **1. Abstract**

Expense Sharing is a web app designed to fulfil the needs of the user by reducing their efforts for the settlement of the bill. The application encourages corresponding users to help in who owes who, and for what. Aim to provide users the best approach to help users and their companion to share expenses easily.

The architecture of our project demands three layers.

- Front end
- Middle layer
- Back end

The front end of the project is handled by “Angular” Framework. The middle layer is built on “SpringBoot” Framework and communicates with mysql database to show the content on the front end via Rest Api.

## 2. Introduction

### 2.1 Overview

In the past, splitting the cost of bills and expenses was a clumsy process. You had to calculate who owes what, then go through the tedious task of collecting money from each person. And since fewer people are carrying cash less often these days, settling up gets even more difficult. That is, until an expense-sharing web app.

Expense Sharing web app lets you divide up the cost of things like group dinners and group trips so that everyone can pay their fair share. With this application, you'll be able to keep the peace between friends by cutting down on squabbles about who owes what to whom.

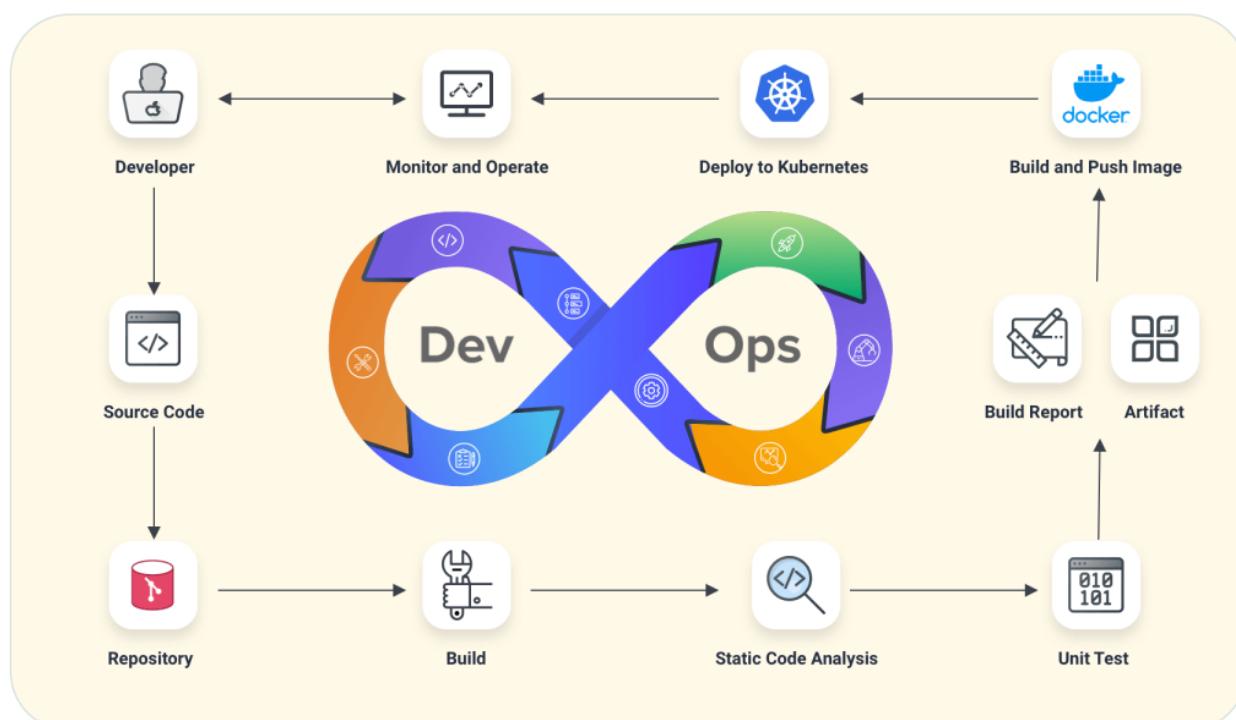
### 2.2 Features

- Users can create a particular group with friends and family and can divide expenses among the members of the group.
- Users can keep track of shared expenses, balances, and who owes who.
- Split expenses with any group: trips, housemates, friends, and family.
- Users can quickly add expenses on the go before you forget who paid.
- Settle up with a friend and record any cash or online payment.

## 2.3 Why DevOps

DevOps is a set of practices that automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably. The concept of DevOps is founded on building a culture of collaboration between teams that historically functioned in relative siloes. The promised benefits include increased trust, faster software releases, ability to solve critical issues quickly, and better manage unplanned work.

It unites agile, continuous delivery, automation, and much more, to help development and operations teams be more efficient, innovate faster, and deliver higher value to businesses and customers. Thus DevOps is preferred as it helps in faster release time and better management of unplanned work.



Figure(1)- DevOps Model



## **3. System Configuration**

### **3.1 Operating System**

Ubuntu 20.04 LTS

### **3.2 CPU and RAM**

4 core processor and RAM 8 GB

### **3.3 Language**

JavaScript XML, JSX, Angular web Framework  
Java, Spring framework

### **3.4 Kernel Version**

Linux Machine 5.13.0-40-generic

### **3.5 Database**

MySQL 8.0.29 (MySQL Community Server - GPL)

### **3.6 Building Tools**

Maven to build java application..

### **3.7 DevOps Tools**

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

# 4. Software Development Life Cycle

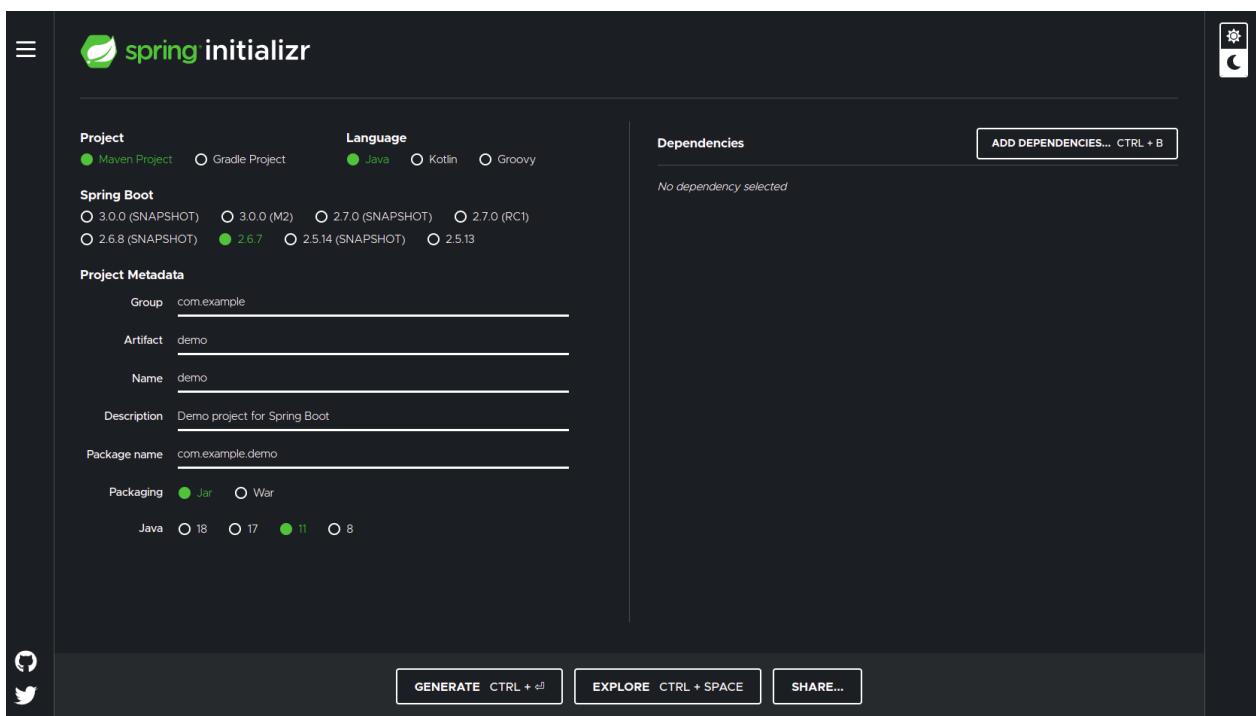
Software development life cycle is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the pre-defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life cycle has its own process and deliverables that feed into the next phase.

## 4.1 Installation

### 4.1.1 SpringBoot

Start Project.

Visit (<https://start.spring.io/>) → Set requirements → Download Project.



Figure(2)- SpringBoot Project download

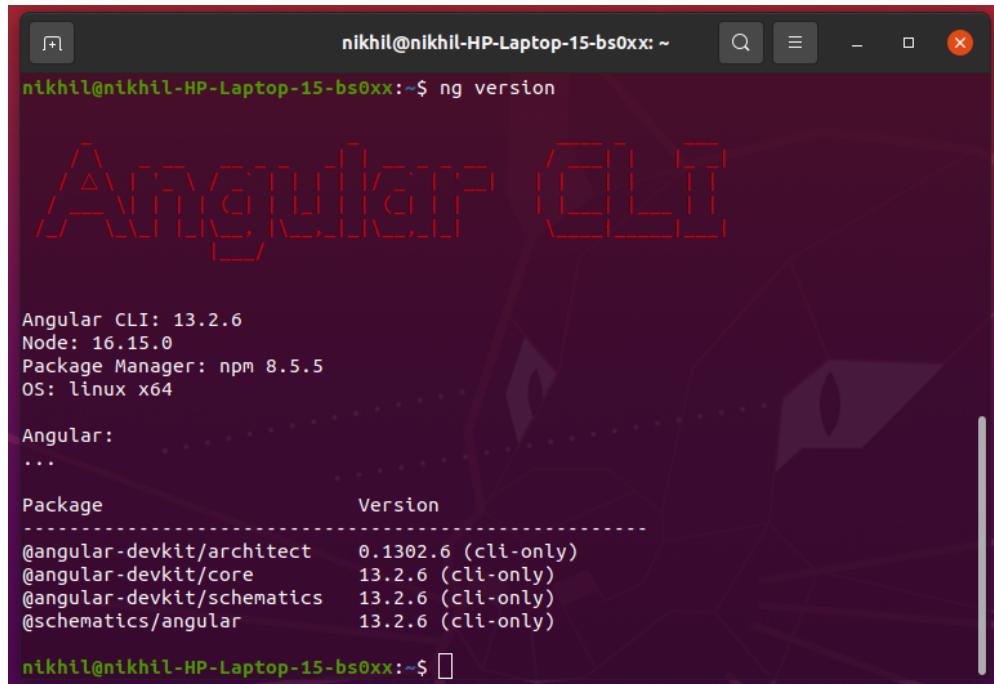
## 4.1.2 Angular

Angular is written in TypeScript. We need Node and npm to compile the files into JavaScript after that, we can deploy our application. For this purpose, Node.js must be installed in the system.

To check → (node - -version)

npm is used to install Angular 8 CLI. Once Node.js is installed, npm is also installed.

```
npm install -g @angular/cli
```



```
nikhil@nikhil-HP-Laptop-15-bs0xx:~$ ng version

Angular CLI: 13.2.6
Node: 16.15.0
Package Manager: npm 8.5.5
OS: linux x64

Angular:
...
Package          Version
-----
@angular-devkit/architect    0.1302.6 (cli-only)
@angular-devkit/core         13.2.6 (cli-only)
@angular-devkit/schematics   13.2.6 (cli-only)
@schematics/angular          13.2.6 (cli-only)

nikhil@nikhil-HP-Laptop-15-bs0xx:~$
```

Figure(3)- Angular Installation

### **4.1.3 MySQL**

MySQL is an open-source relational database management system.

#### **Installing MySQL**

- sudo apt update
- sudo apt install mysql-server
- sudo mysql\_secure\_installation

#### **Configuring MySQL**

- sudo mysql\_secure\_installation

This will take you through a progression of prompts where you can roll out certain improvements to your MySQL establishment security alternatives.

### **4.1.4 Docker**

Docker is a continuous delivery tool. In a business enterprise, requirements keep changing and new updates keep waiting for deployment. The process of deployment has various prerequisite steps that need to be followed.

After installing some prerequisite packages, we can install docker.

#### **Installing Docker**

- apt install docker-ce
- usermod -aG docker \${USER}
- su - \${USER}

## 4.2 Source Control Management

We have used Git for source code management. A git repository is created on the Github with the team members as the collaborators. Hence, any of the members can push the changed code to the repository. Github helped us to work in a synchronous manner so we can change or access the code at the same time. IntelliJ provides VCS (version control system) to handle git commands and connects the local repository to the remote Github repository.

1. git init:- This command is used to create a new blank repository. It is used to make an existing project as a Git project.
2. git clone <url>:- This command is used to clone the remote repository to local system.
3. git add :- The git add command is used to add file contents to the Index(Staging Area) .This command updates the current content of the working tree to the staging area.
4. git commit -m <commit message>:- This creates a snapshot of the files and commits them by creating a SHA id.
5. git remote add origin <github-repository-url>:- This creates a new remote called origin located at given url.
6. git push -u origin <branch>:- This command is used for pushing the local changes to repository on server on specified branch.

Github repository links for the project:-

1. <https://github.com/bolleyboll/Splitwise-API.git>
2. <https://github.com/bolleyboll/Splitwise.git>

main ▾ 1 branch 0 tags

Go to file Add file Code

**bolleyball Minor Fixes**

9668ac9 4 hours ago 24 commits

|  |  |  |
| --- | --- | --- |
| src | Minor Fixes | 4 hours ago |
| .gitignore | Boilerplate | 5 days ago |
| AnsibleDeploy.yml | stopping already running docker compose | 16 hours ago |
| Dockerfile | dockerfile, ansible playbook, docker-compose, inventory path updated | 21 hours ago |
| Jenkinsfile | name of jenkins file added | 14 hours ago |
| LICENSE | Initial commit | 5 days ago |
| README.md | Boilerplate | 5 days ago |
| angular/README.md | Boilerplate | 5 days ago |
| docker-compose.yml | compose file updated | 21 hours ago |
| inventory | inventory updated | 20 hours ago |
| karma.conf.js | Boilerplate | 5 days ago |
| ngsw-config.json | Boilerplate | 5 days ago |
| package-lock.json | Boilerplate | 5 days ago |
| package.json | Boilerplate | 5 days ago |
| tsconfig.app.json | Boilerplate | 5 days ago |
| tsconfig.json | Registration Done + Login | 2 days ago |
| tsconfig.spec.json | Boilerplate | 5 days ago |

Figure(4)- Github repository for backend

main ▾ 1 branch 0 tags

Go to file Add file Code

**bolleyball Adding Members Done**

8de16d5 22 minutes ago 32 commits

|  |  |  |
| --- | --- | --- |
| .mvn/wrapper | BoilerPlate Code | 5 days ago |
| .vscode | Logger | 2 days ago |
| src | Adding Members Done | 22 minutes ago |
| .gitignore | Props | 21 hours ago |
| AnsibleDeploy.yml | Jenkins pipeline added | 16 hours ago |
| Dockerfile | Final Dockerfile | 3 hours ago |
| HELP.md | BoilerPlate Code | 5 days ago |
| Jenkinsfile | Jenkins pipeline added | 16 hours ago |
| LICENSE | Initial commit | 5 days ago |
| README.md | Initial commit | 5 days ago |
| docker-compose.yml | Final Dockerfile | 3 hours ago |
| inventory | Jenkins pipeline added | 16 hours ago |
| mvnw | BoilerPlate Code | 5 days ago |
| mvnw.cmd | BoilerPlate Code | 5 days ago |
| pom.xml | Logging + Testing | yesterday |

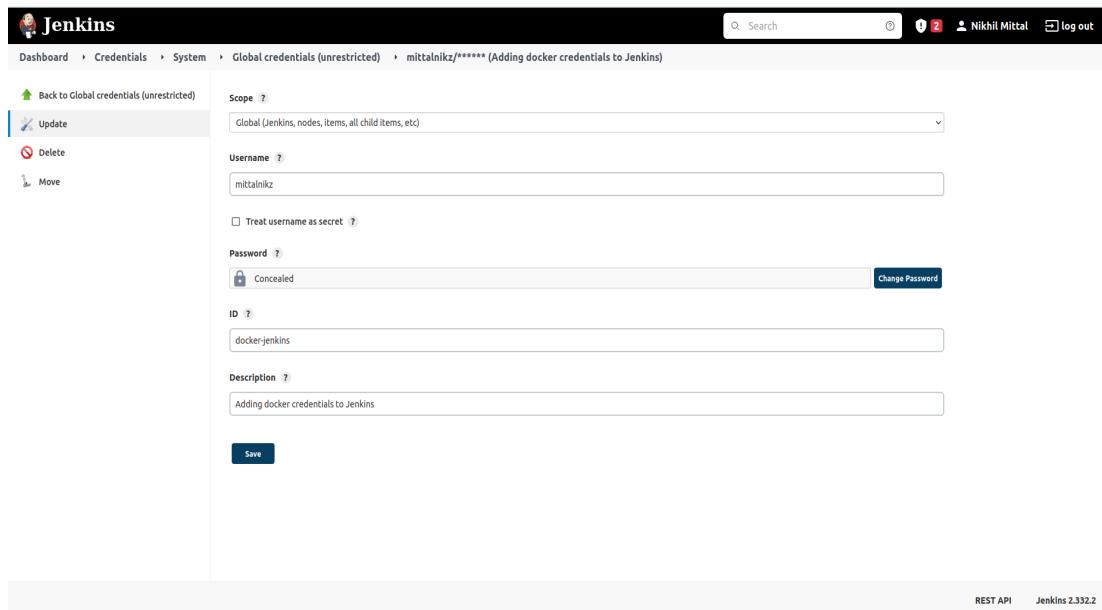
README.md

Figure(5)- Github repository for backend

## 4.2.1 CI Jenkins Pipeline:-

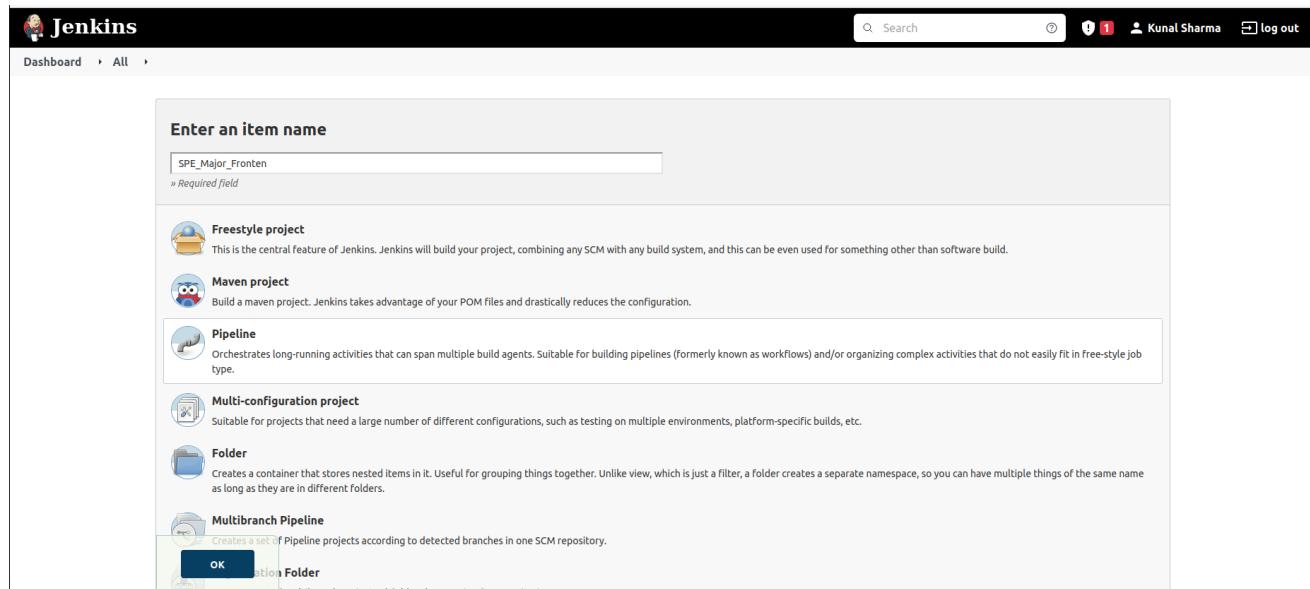
We used Jenkins Pipeline to clone these repositories from git.

- First we add credentials of Github repository



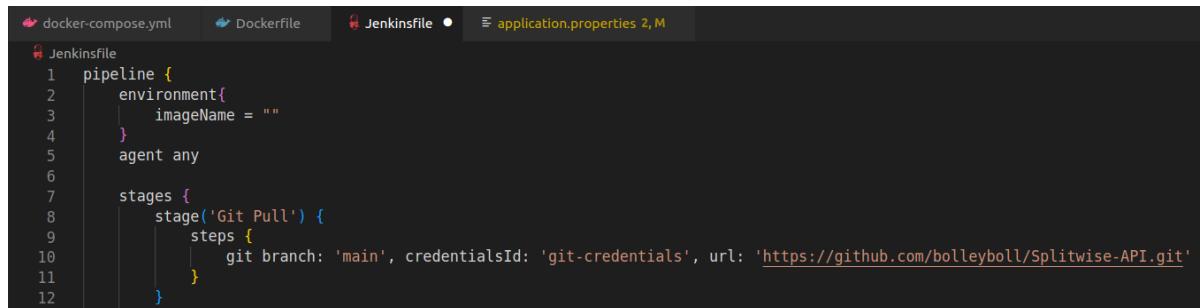
Figure(6)- Adding Github credentials

- Then we create a Jenkins pipeline.



Figure(7)- Jenkins pipeline.

## ➤ Git Script with jenkins pipeline



```
❶ Jenkinsfile
1 pipeline {
2     environment{
3         |     imageName = ""
4     }
5     agent any
6
7     stages {
8         |     stage('Git Pull') {
9             |         steps {
10                |             git branch: 'main', credentialsId: 'git-credentials', url: 'https://github.com/bolleyboll/Splitwise-API.git'
11            }
12        }
13    }
14 }
```

Figure(8)- Pipeline Script

## 4.3 Build

### 4.3.1 Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting, and documentation from a central piece of information. It handles dependencies and plugins so we don't need to add jar files for the library explicitly. First, it checks for project dependencies in pom.xml and then compiles the code for any errors.

We have to install maven in our system so we can build what we write. To install Maven we want to first install its dependency JAVA, here we will be installing java version 8.

```
sudo apt update
sudo apt install openjdk-8-jdk
```

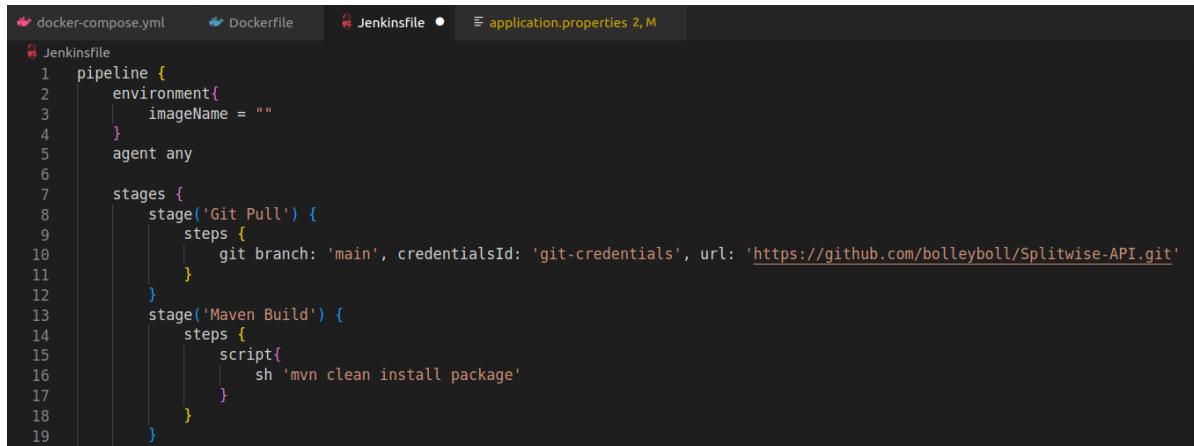
We will now install maven on our setup

```
sudo apt update
sudo apt install maven
```

## 4.3.2 CI Pipeline

Jenkins uses the system's maven to build the maven project, so we must have the maven integration plugin installed on our Jenkins.

- Install Maven integration plugin
- We add another stage to our pipeline.



```
docker-compose.yml  Dockerfile  Jenkinsfile  application.properties 2, M
Jenkinsfile
1 pipeline {
2     environment{
3         imageName = ""
4     }
5     agent any
6
7     stages {
8         stage('Git Pull') {
9             steps {
10                 git branch: 'main', credentialsId: 'git-credentials', url: 'https://github.com/bolleyball/Splitwise-API.git'
11             }
12         }
13         stage('Maven Build') {
14             steps {
15                 script{
16                     sh 'mvn clean install package'
17                 }
18             }
19         }
20     }
21 }
```

Figure(9)- Maven build script

- In mvn clean install, it also runs all the test cases for the project and creates the executable jar file in the target folder in the current working directory.

## 4.4 Testing

Jenkins provides us with continuous integration which includes integrated testing, so every time we push code to github it integrates all the different modules of the project and tests their proper functioning.

JUnit is a simple, open source framework to write and run repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks.

JUnit promotes the idea of "first testing then coding", which emphasises on setting up the test data for a piece of code that can be tested first and then implemented.

This approach is like "test a little, code a little, test a little, code a little." It increases the productivity of the programmer and the stability of program code, which in turn reduces the stress on the programmer and the time spent on debugging.

```
src > test > java > org > iiitb > splitwise > SplitwiseApplicationTests.java > ...
1 package org.iiitb.splitwise;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.test.context.SpringBootTest;
6 import org.springframework.boot.test.web.client.TestRestTemplate;
7 import org.springframework.boot.web.server.LocalServerPort;
8 import org.springframework.boot.test.context.SpringBootTest.WebEnvironment;
9 import static org.assertj.core.api.Assertions.assertThat; You, yesterday • Logging
10
11 You, yesterday | 1 author (You)
12 @SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
13 class SplitwiseApplicationTests {
14
15     @LocalServerPort
16     private int port;
17
18     @Autowired
19     private TestRestTemplate restTemplate;
20
21     @Test
22     void contextLoads() {
23
24         @Test
25         public void greetingShouldReturnDefaultMessage() throws Exception {
26             assertThat(this.restTemplate.getForObject("http://localhost:" + port + "/",
27                 | | | responseType: String.class)).contains(...values: ""));
28         }
29
30 }
```

Figure(10)- Test Cases

For testing within java code we use Junit. Junit test cases are written in covering the scope of the project. Also Junit dependency is added in pom.xml for maven to resolve the dependency.

## 4.5 Docker Artifact

Docker is an application that makes it simple and easy to run application processes in a container, which are like virtual machines, only more portable, more resource-friendly, and more dependent on the host operating system.

Artifact repository tools store all of the bulk binary artifact files that rarely, if ever, need to be altered or changed. Docker offers a tool like this.

First, we need to install docker on our system. Once it is installed, we can run it using following command:

```
$ sudo systemctl start docker
```

Here, our aim is to build a docker image out of the jar file that will be created in build phase and push the latest image to Docker Hub, which will then be pulled by Ansible to deploy it to other machines. All this will be automated in Jenkins pipeline. To push the docker image, use the following command:

```
docker push <username>/<repository_name>:tagname
```

To pull the docker image, use the following command:

```
docker pull <docker_image_name:tag>
```

To run the docker image, use the following command:

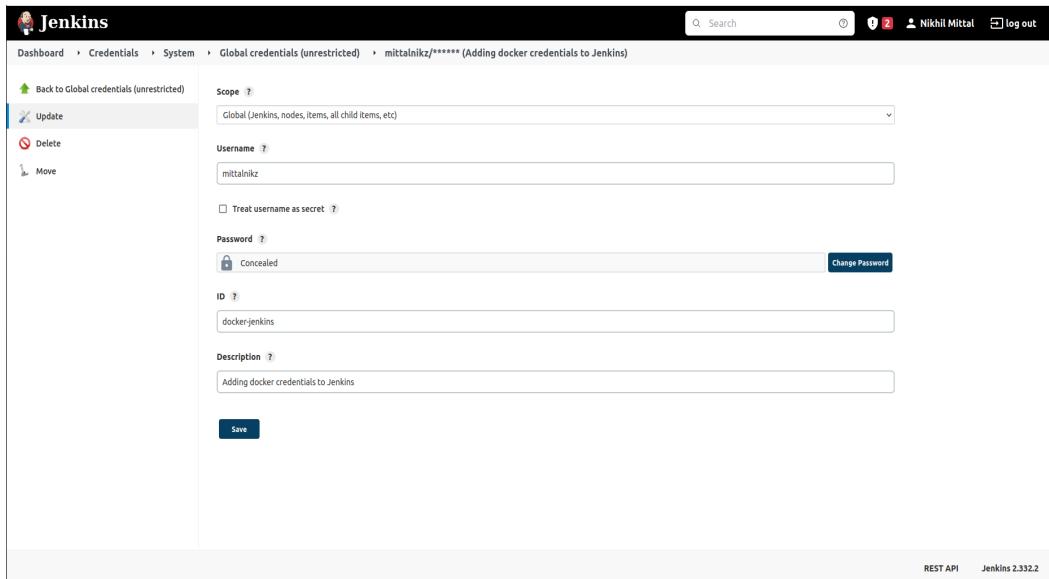
```
docker run -it <image_name>
```

Also, make sure to add Jenkins to docker group, so that Jenkins can use docker for build docker image. This can be done by following command:

```
$ sudo usermod -aG docker Jenkins
```

Docker hub links for the project:-

1. <https://hub.docker.com/r/patro30/frontend>
2. <https://hub.docker.com/r/patro30/backend>



Figure(11)- Adding Docker credentials

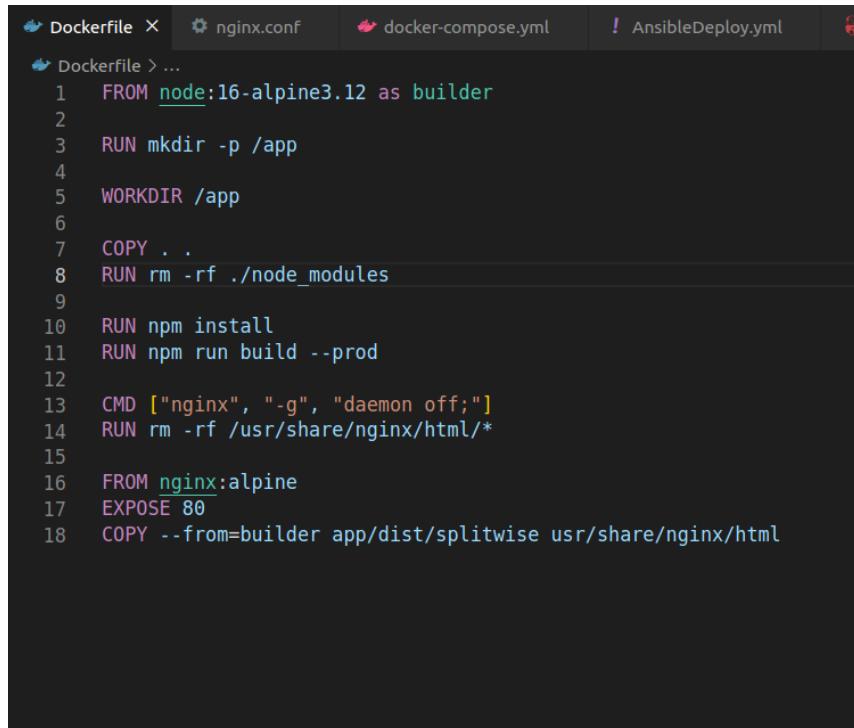
#### 4.5.1 CI Pipeline

- Add docker integration plugin for docker in Jenkins under Jenkins → Plugin Manager.
- Build and Push Spring Boot Image.

```
FROM openjdk:11
ADD target/splitwise-0.0.1-SNAPSHOT.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Figure(12)- Dockerfile for frontend

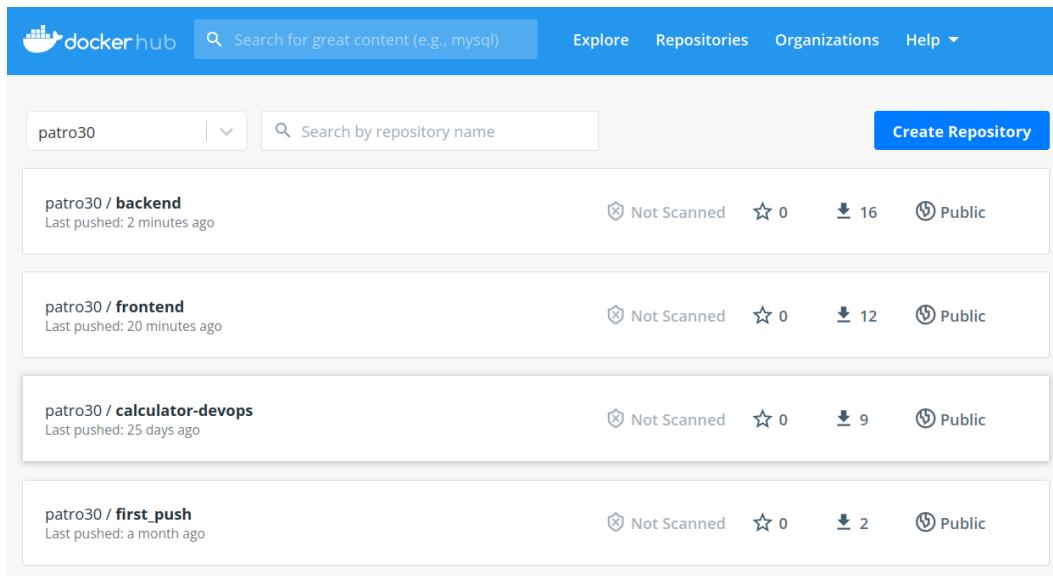
➤ Build and Push Angular Image.



```
FROM node:16-alpine3.12 as builder
RUN mkdir -p /app
WORKDIR /app
COPY . .
RUN rm -rf ./node_modules
RUN npm install
RUN npm run build --prod
CMD ["nginx", "-g", "daemon off;"]
RUN rm -rf /usr/share/nginx/html/*
FROM nginx:alpine
EXPOSE 80
COPY --from=builder app/dist/splitwise /usr/share/nginx/html
```

Figure(13)- Dockerfile for backend

➤ After a successful deployment of docker images, we can see the result on our docker-hub account.



Figure(14)- DockerHub dashboard

## 4.5.2 Docker compose

Docker Compose is a tool for defining and running multi-container Docker applications.

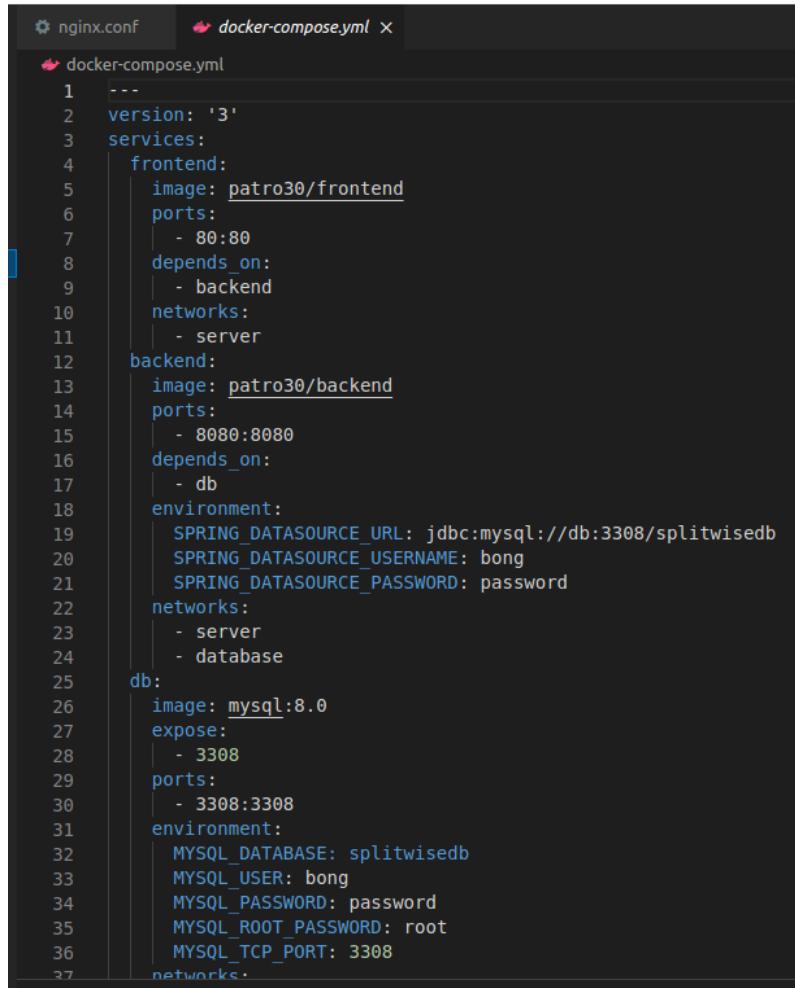
With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

Docker Compose commands :

**Docker-compose up :** Start all services ( it will run docker-compose.yml file )

**Docker-compose build :** This command will build the images with the help of docker-compose.yml file.

**Docker-compose down :** Stop all the services.



```
nginx.conf docker-compose.yml
docker-compose.yml
1 ---
2   version: '3'
3   services:
4     frontend:
5       image: patro30/frontend
6       ports:
7         - 80:80
8       depends_on:
9         - backend
10      networks:
11        - server
12    backend:
13      image: patro30/backend
14      ports:
15        - 8080:8080
16      depends_on:
17        - db
18      environment:
19        SPRING_DATASOURCE_URL: jdbc:mysql://db:3308/splitwisedb
20        SPRING_DATASOURCE_USERNAME: bong
21        SPRING_DATASOURCE_PASSWORD: password
22      networks:
23        - server
24        - database
25    db:
26      image: mysql:8.0
27      expose:
28        - 3308
29      ports:
30        - 3308:3308
31      environment:
32        MYSQL_DATABASE: splitwisedb
33        MYSQL_USER: bong
34        MYSQL_PASSWORD: password
35        MYSQL_ROOT_PASSWORD: root
36        MYSQL_TCP_PORT: 3308
37      networks:
```

Figure(15)- Docker compose file

## 4.6 Continuous Deployment

A Deployment pipeline is the process of taking code from version control and making it readily available to users of your application in an automated fashion. When a team of developers are working on projects or features they need a reliable and efficient way to build, test and deploy their work.

We are using Ansible for the deployment. It is a system of configuration management written in Python programming language which uses a declarative markup language to describe configurations. It's used for automation of configuration and OS setup. It is often used to manage Linux-nodes, but Windows is also supported.

Ansible allows us to write 'Playbooks' that are descriptions of the desired state of our systems, which are usually kept in source control. Ansible then does the hard work of getting your systems to that state no matter what state they are currently in. Playbooks make your installations, upgrades, and day-to-day management repeatable and reliable.

First, we need to install ansible on our system. Also, Ansible in the backend uses python to run various small codes and uses ssh in linux to connect to the hosts machines. So, we need to make sure that python(generally installed in linux) and ssh server are installed.

We can install them using following commands:

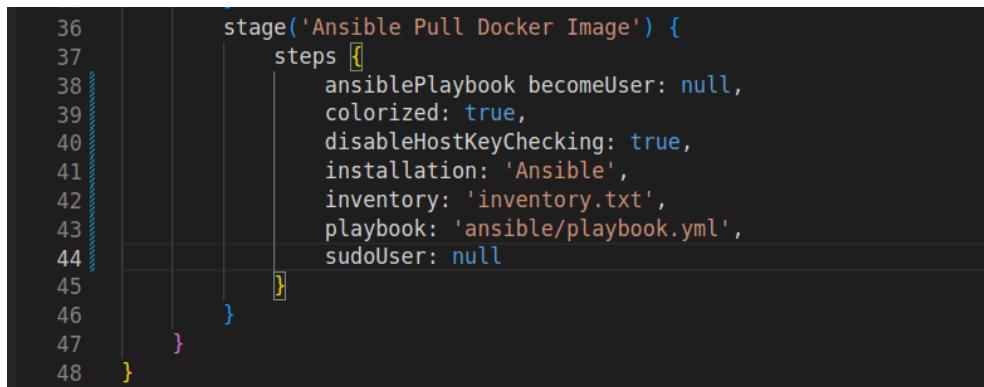
```
$ sudo apt install openssh-server  
$ ssh-keygen -t rsa  
$ sudo apt update  
$ sudo apt install ansible
```

After the ansible is installed, we can verify it by checking its version using

```
$ ansible -version
```

## 4.6.1 CI Pipeline

- Jenkins would call Ansible to deploy the docker image on the deployment node.

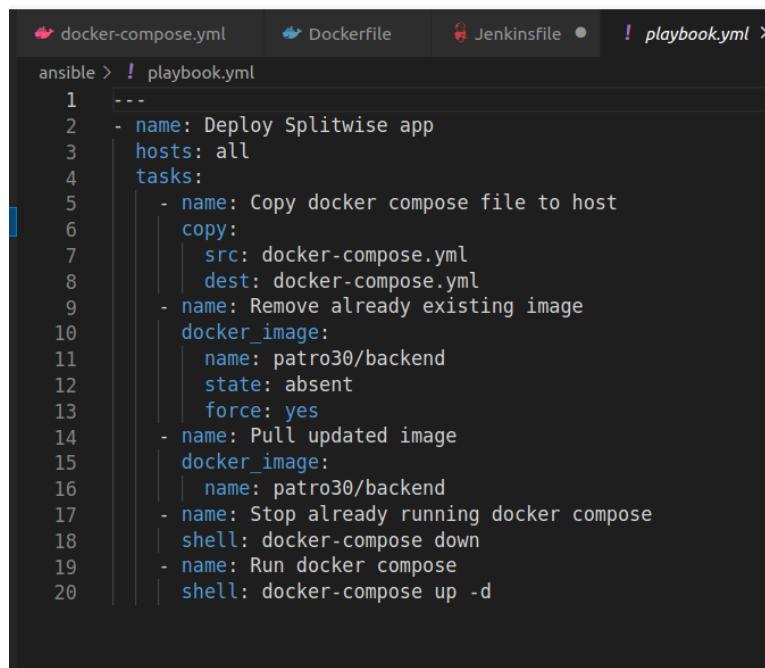


```
36     stage('Ansible Pull Docker Image') {
37         steps [
38             ansiblePlaybook becomeUser: null,
39             colorized: true,
40             disableHostKeyChecking: true,
41             installation: 'Ansible',
42             inventory: 'inventory.txt',
43             playbook: 'ansible/playbook.yml',
44             sudoUser: null
45         ]
46     }
47 }
48 }
```

Figure(16)- Ansible pipeline script

- Ansible uses YAML syntax for expressing Ansible playbooks.

YAML uses simple key-value pairs to represent the data. The dictionary is represented in key: value pair.



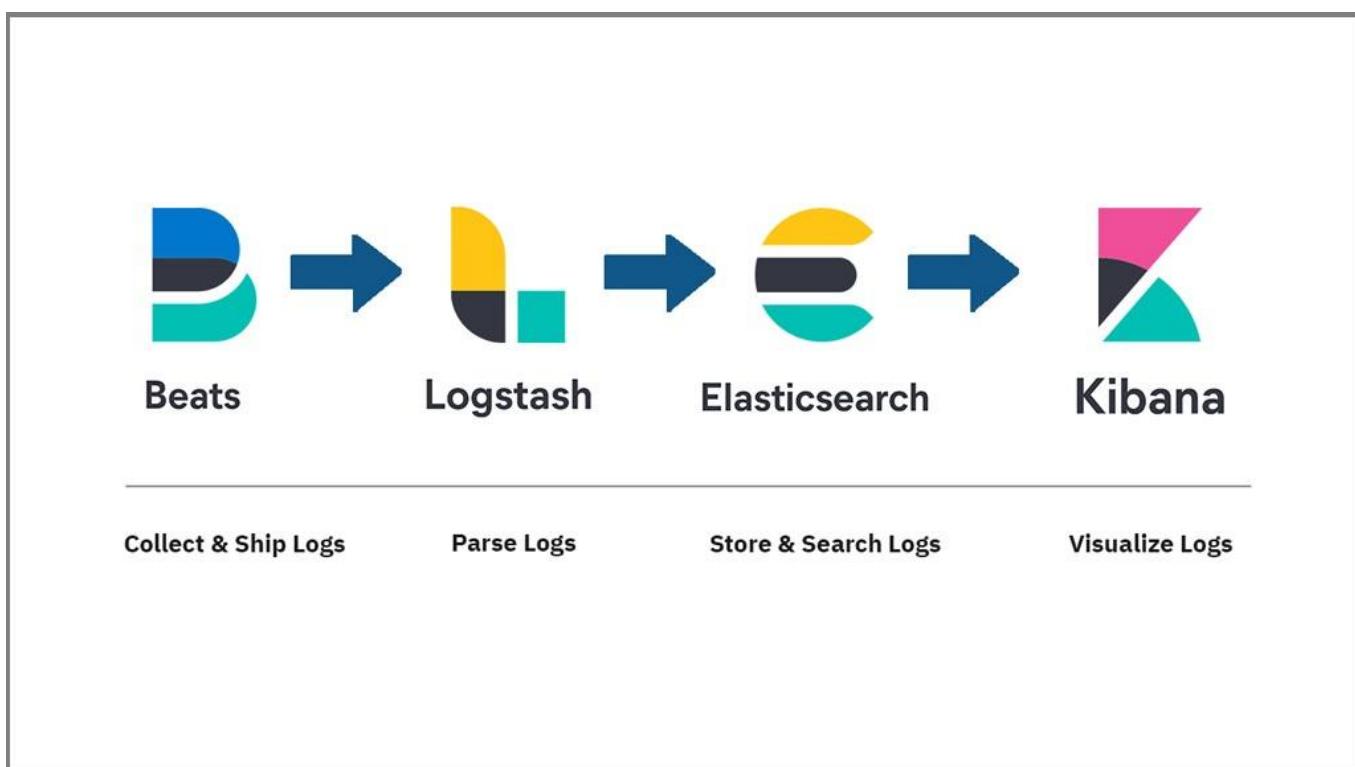
```
ansible > playbook.yml
1 ...
2 - name: Deploy Splitwise app
3   hosts: all
4   tasks:
5     - name: Copy docker compose file to host
6       copy:
7         src: docker-compose.yml
8         dest: docker-compose.yml
9     - name: Remove already existing image
10       docker_image:
11         name: patro30/backend
12         state: absent
13         force: yes
14     - name: Pull updated image
15       docker_image:
16         name: patro30/backend
17     - name: Stop already running docker compose
18       shell: docker-compose down
19     - name: Run docker compose
20       shell: docker-compose up -d
```

Figure(17)- playbook file

## 4.7 Monitor

The final stage in the DevOps paradigm is monitoring.

In this step, developers keep track of user interaction and another usage of the application accordingly. The architecture used for monitoring is the 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 visualise data with charts and graphs in Elasticsearch.

The ELK Stack is famous in light of the fact that it satisfies a need in the log, the executives and investigation space.

In addition to adding integrations, you can try our sample data or upload your own data.

[Sample data](#) [Upload file](#)

### Visualize data from a log file

Upload your file, analyze its data, and optionally import the data into an Elasticsearch index.

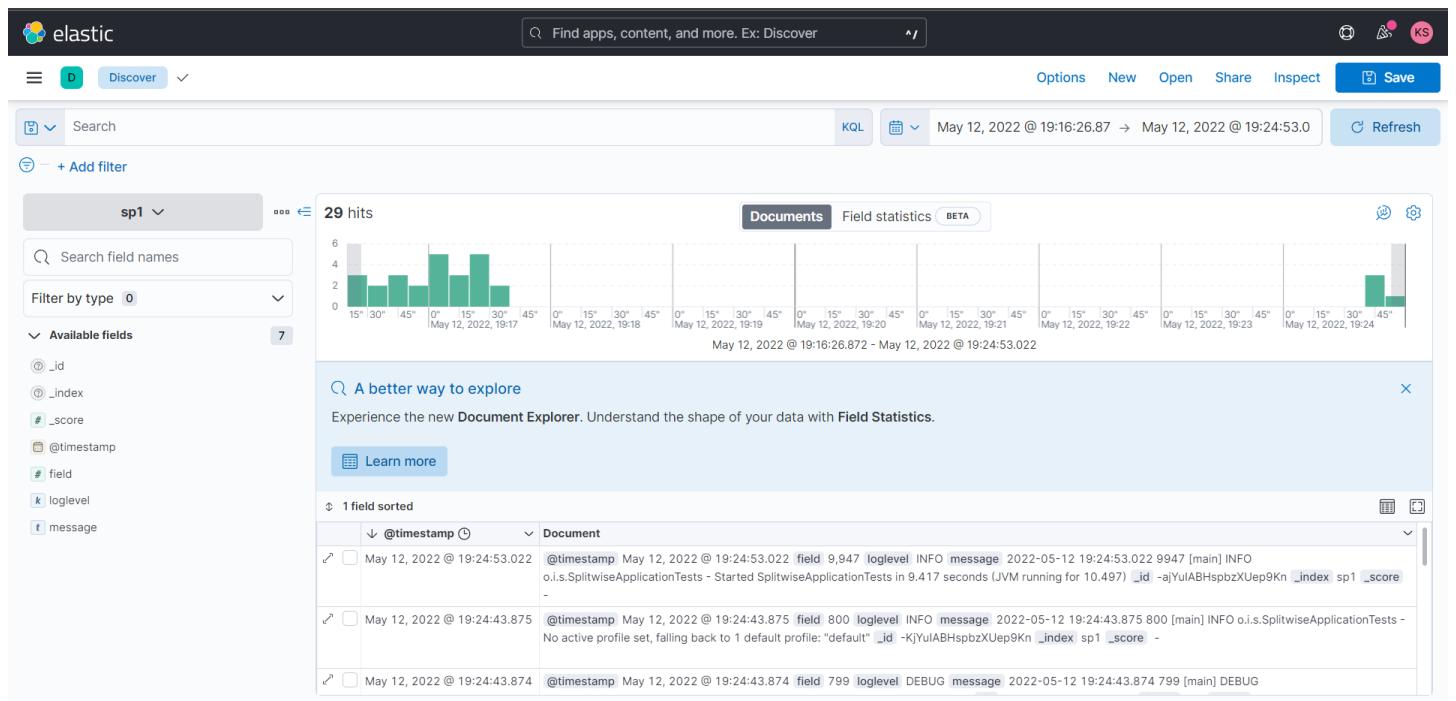
The following file formats are supported:

- Delimited text files, such as CSV and TSV
- Newline-delimited JSON
- Log files with a common format for the timestamp

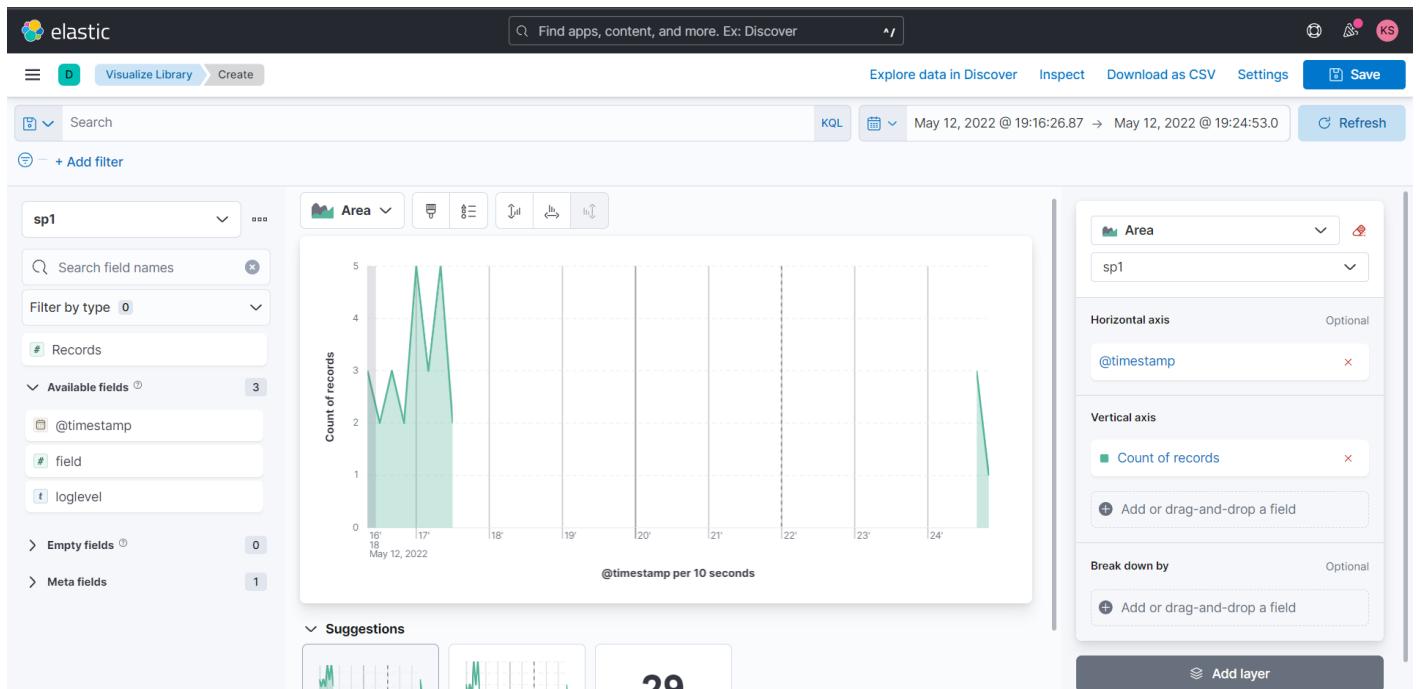
You can upload files up to 100 MB.

Select or drag and drop a file

Figure(18)- uploading log file



Figure(19)- Kibana visualisation

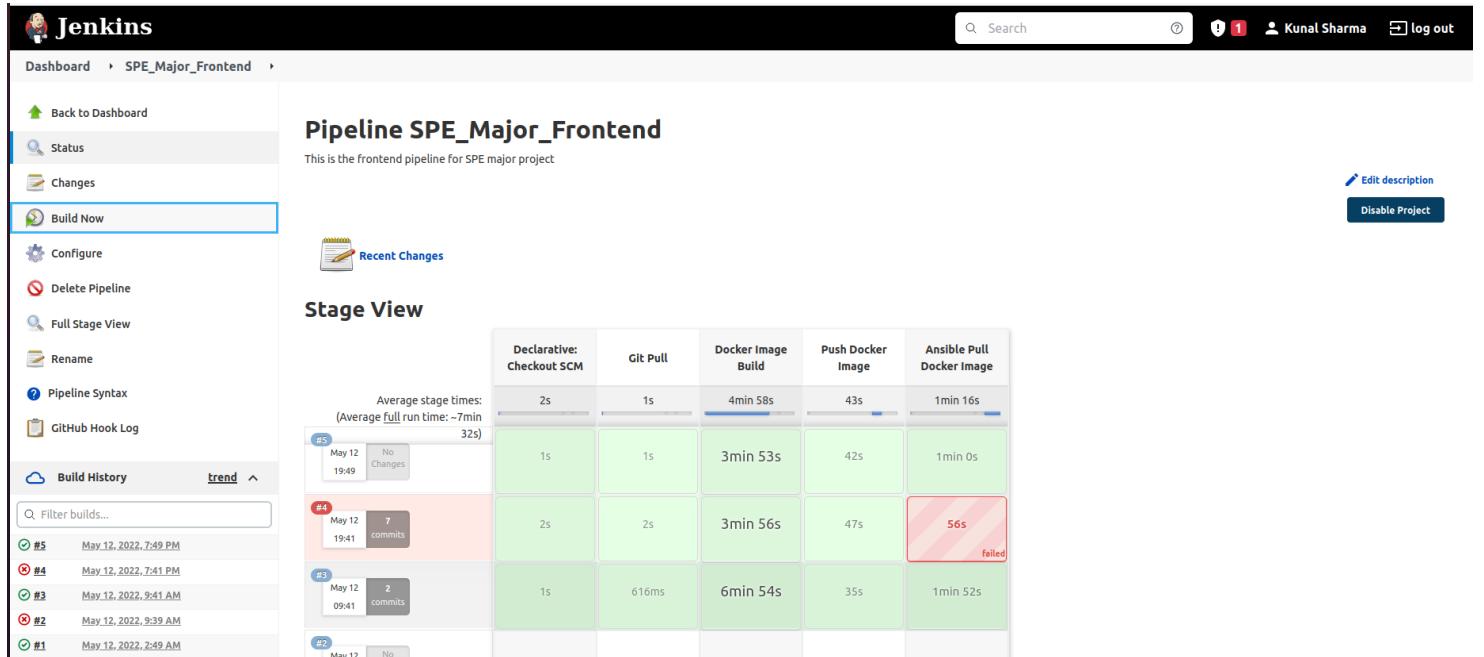


Figure(20)- Kibana visualisation

## 4.8 Building Pipeline and Running on Deployed Machine

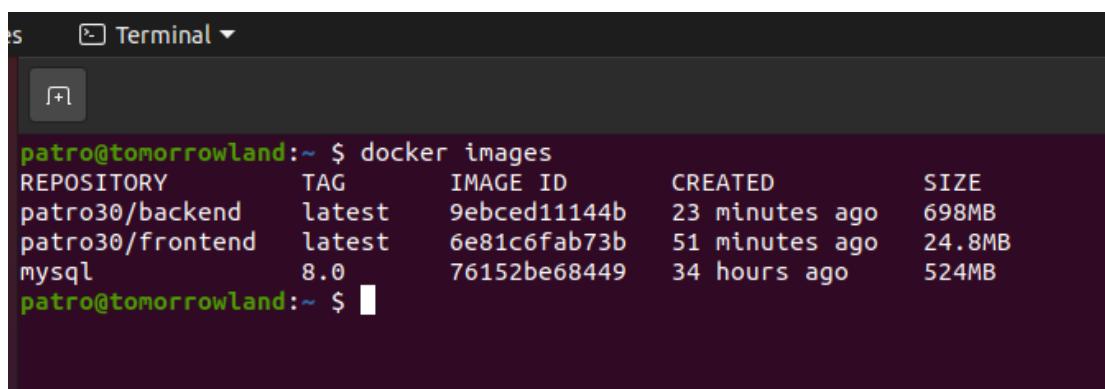
After completing all stages of SDLC we deployed the images successfully to the deployment node. We pulled the repo from git, build from maven , build and deploy the docker images and at last configured Ansible with Jenkins.

- Pipeline stages in Jenkins.



Figure(21)- Pipeline stages

- Deployed images on the target machine.



```
patro@tomorrowland:~ $ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
patro30/backend latest   9ebced11144b  23 minutes ago  698MB
patro30/frontend latest   6e81c6fab73b  51 minutes ago  24.8MB
mysql           8.0     76152be68449  34 hours ago   524MB
patro@tomorrowland:~ $
```

Figure(22)- Images on target machine

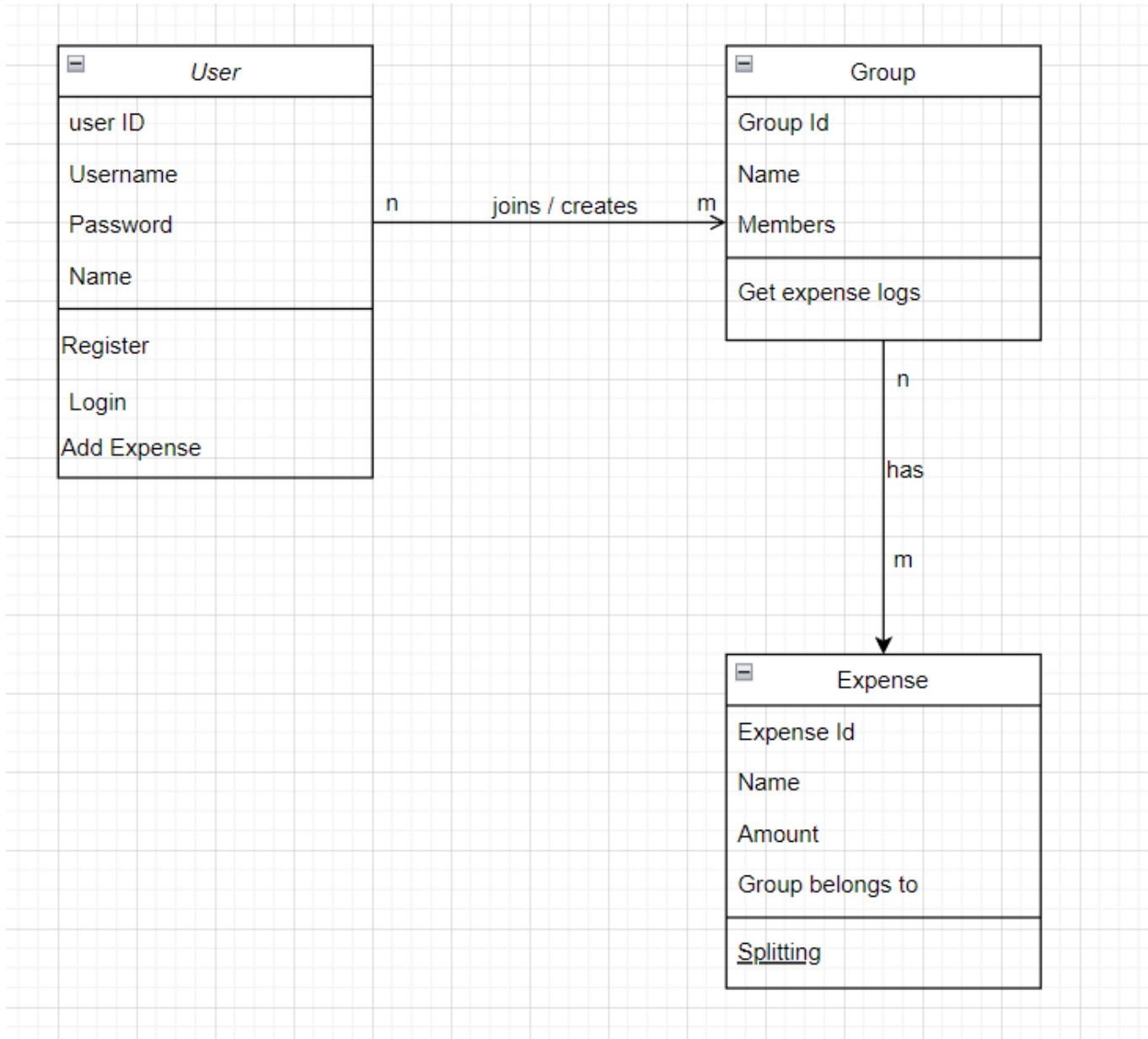
---

## 5. Experimental Setup

### 5.1 Functional Requirements

- Users can sign in into their account and can see all their previous expenses and bill splits.
- Users can create a particular group with friends and family and can divide expenses among the members of the group.
- Users can keep track of shared expenses, balances, and who owes who.
- Split expenses with any group: trips, housemates, friends, and family.
- Users can quickly add expenses on the go before you forget who paid.
- Settle up with a friend and record any cash or online payment.

## 5.2 Class Diagram



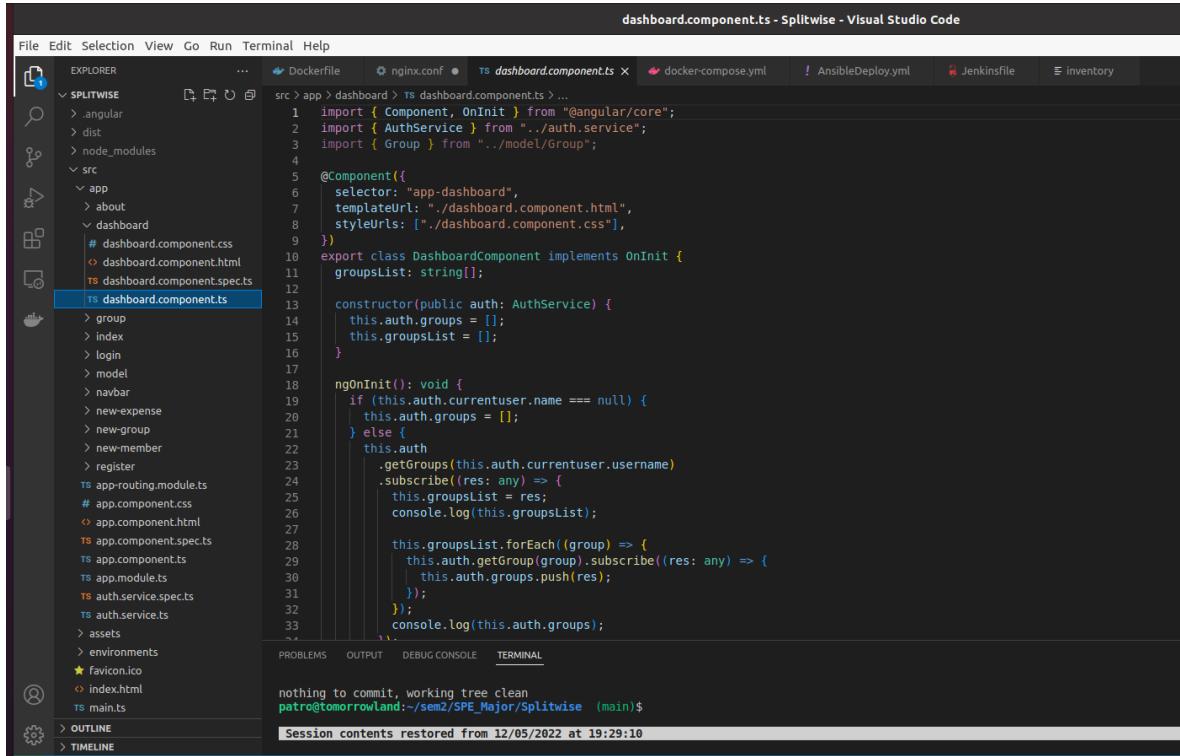
Figure(23)- Class diagram

## 5.3 API Documentation

| S.no | Use Case                                    | API                           | Request Method | Role  |
|------|---|-------------------------------|----------------|-------|
| 1    | User Registration                           | /register                     | POST           | USER  |
| 2    | User Login                                  | /login                        | POST           | USER  |
| 3    | Get list of all the users                   | /users                        | GET            | ADMIN |
| 4    | Adding the expense to the group             | /expense/add                  | POST           | USER  |
| 5    | Get the group details                       | /group/{id}                   | GET            | USER  |
| 6    | Creating a group                            | /group/create                 | POST           | USER  |
| 7    | Get the expense details of particular group | /group/expenses/{id}          | GET            | USER  |
| 8    | Adding another user to the group            | /group/add/{username}/{grpld} | PUT            | USER  |
| 9    | Get all the groups for particular user      | /user/group/{username}        | GET            | USER  |

## 5.4 Code Walkthrough

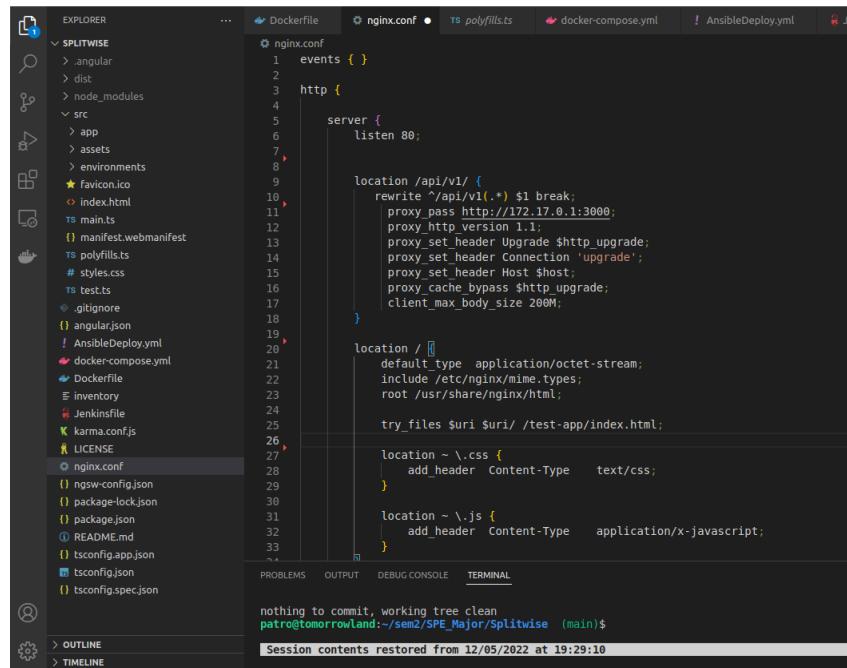
- Below code shows the modules that we have used in the front-end and components that we have created to drive our app.



The screenshot shows the Visual Studio Code interface with the file `dashboard.component.ts` open in the editor. The code implements an `OnInit` interface and uses Angular services like `AuthService` and `Group`. The code is as follows:

```
1 import { Component, OnInit } from '@angular/core';
2 import { AuthService } from './auth.service';
3 import { Group } from '../model/Group';
4
5 @Component({
6   selector: "app-dashboard",
7   templateUrl: "./dashboard.component.html",
8   styleUrls: ['./dashboard.component.css'],
9 })
10 export class DashboardComponent implements OnInit {
11   groupsList: string[];
12
13   constructor(public auth: AuthService) {
14     this.auth.groups = [];
15     this.groupsList = [];
16   }
17
18   ngOnInit(): void {
19     if (this.auth.currentuser.name === null) {
20       this.auth.groups = [];
21     } else {
22       this.auth
23         .getGroups(this.auth.currentuser.username)
24         .subscribe((res: any) => {
25           this.groupsList = res;
26           console.log(this.groupsList);
27
28           this.groupsList.forEach((group) => {
29             this.auth.getGroup(group).subscribe((res: any) => {
30               this.auth.groups.push(res);
31             });
32           });
33           console.log(this.auth.groups);
34         });
35     }
36   }
37 }
```

Figure(24)- Dashboard Components



The screenshot shows the Visual Studio Code interface with the file `nginx.conf` open in the editor. The configuration defines an `events` block and an `http` block with a `server` block listening on port 80. It includes proxying for a backend at port 3000 and static file serving for the root directory. The code is as follows:

```
1 events { }
2
3 http {
4
5   server {
6     listen 80;
7
8     location /api/v1/ {
9       rewrite ^/api/v1(.*) $1 break;
10      proxy_pass http://172.17.0.1:3000;
11      proxy_http_version 1.1;
12      proxy_set_header Upgrade $http_upgrade;
13      proxy_set_header Connection 'upgrade';
14      proxy_set_header Host $host;
15      proxy_cache_bypass $http_upgrade;
16      client_max_body_size 200M;
17    }
18
19    location / {
20      default_type application/octet-stream;
21      include /etc/nginx/mime.types;
22      root /usr/share/nginx/html;
23
24      try_files $uri $uri/ /test-app/index.html;
25    }
26
27    location ~ \.css {
28      add_header Content-Type text/css;
29    }
30
31    location ~ \.js {
32      add_header Content-Type application/x-javascript;
33    }
34  }
35}
```

Figure(25)- nginx configuration

```

src > app > TS auth.service.ts ...
1  // ...
2
3  @Injectable({
4      providedIn: "root",
5  })
6  export class AuthService {
7      currentUser: User = null;
8      isLoggedIn: Boolean = false;
9      currentgroup: string;
10     curGrp: Group;
11     groups: Group[];
12
13     constructor(public http: HttpClient) {}
14
15     getExpenses(id) {
16         return this.http.get("http://localhost:8080/group/expenses/" + id);
17     }
18     signIn(login) {
19         return this.http.post("http://localhost:8080/login", login);
20     }
21     patRegister(user) {
22         return this.http.post("http://localhost:8080/register", user);
23     }
24     groupRegister(grp) {
25         return this.http.post("http://localhost:8080/group/create", grp);
26     }
27     newExpense(exp) {
28         return this.http.post("http://localhost:8080/expense/add", exp);
29     }
30     getAllUsers() {
31         return this.http.get("http://localhost:8080/users");
32     }
33     // showVaccines() {
34     //     return this.http.get("http://localhost:8080/vaccines");
35 }
36
37
38
39
40
41
42
43
44

```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

nothing to commit, working tree clean  
patro@tomorrowland:~/sem2/SPE\_Major/Splitwise (main)\$  
Session contents restored from 12/05/2022 at 19:29:10

```

src > app > navbar > TS navbar.component.ts ...
1  import { Component, OnInit } from '@angular/core';
2  import { AuthService } from '../auth.service';
3  import { User } from '../model/User';
4
5  @Component({
6      selector: 'app-navbar',
7      templateUrl: './navbar.component.html',
8      styleUrls: ['./navbar.component.css'],
9  })
10 export class NavbarComponent {
11     constructor(public auth: AuthService) {}
12     logout() {
13         this.auth.currentUser = null;
14         this.auth.isLoggedIn = false;
15     }
16 }

```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

nothing to commit, working tree clean  
patro@tomorrowland:~/sem2/SPE\_Major/Splitwise (main)\$  
Session contents restored from 12/05/2022 at 19:29:10

Figure(26)- Navbar component

```

src > main > java > org > iiitb > splitwise > controller > AppController.java Language Support for Java(TM) by Red Hat > {} org.iiitb.splitwise.controller
6
7 import org.iiitb.splitwise.model.Group;
8 import org.iiitb.splitwise.model.User;
9 import org.iiitb.splitwise.services.UserService;
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.web.bind.annotation.CrossOrigin;
13 import org.springframework.web.bind.annotation.GetMapping;
14 import org.springframework.web.bind.annotation.PathVariable;
15 import org.springframework.web.bind.annotation.PostMapping;
16 import org.springframework.web.bind.annotation.RequestBody;
17 import org.springframework.web.bind.annotation.RestController;
18
19 import lombok.extern.slf4j.Slf4j;
20
21 @CrossOrigin
22 @RestController
23 @Slf4j
24 public class AppController {
25
26     @Autowired
27     private UserService us;
28
29     @Autowired
30     private HttpSession httpSession;
31
32     @PostMapping("register")
33     public ResponseEntity<String> register(@RequestBody User user) {
34         User u = us.saveUser(user);
35         if (u != null) {
36             log.info("[200] - [ " + u.getUsername() + " logged in]");
37             return ResponseEntity.ok(u.getUsername());
38         } else {
39             return ResponseEntity.ok("User already exists!");
40         }
41     }
42
43     @PostMapping("login")
44     public ResponseEntity<User> login(@RequestBody User user) {
45         User dbUsr = us.login(user.getEmail(), user.getPassword());
46

```

Figure(27)- App controller

```

src > main > java > org > iiitb > splitwise > services > impl > ExpenseServiceImpl.java Language Support for Java(TM) by Red Hat > {} org.iiitb.splitwise.services.impl
6 import org.iiitb.splitwise.repositories.GroupRepository;
7 import org.iiitb.splitwise.services.ExpenseService;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.stereotype.Service;
10 import org.springframework.transaction.annotation.Transactional;
11
12 @Service
13 @Transactional
14 public class ExpenseServiceImpl implements ExpenseService {
15
16     @Autowired
17     private GroupRepository gr;
18
19     @Autowired
20     private ExpenseRepository er;
21
22     @Override
23     public Expense add(Expense expense) {
24         String paidBy = expense.getSplitBetween();
25         Double amt = expense.getAmount();
26         String f = paidBy + ":" + amt.toString() + ", ";
27         Integer grpId = Integer.parseInt(expense.getForGroup());
28         Group g = gr.findById(grpId);
29         String members = g.getMembers();
30         String memL[] = members.split(", ");
31         int count = memL.length;
32         Double split = amt / count;
33
34         for (int i = 0; i < memL.length; i++) {
35             if (memL[i].equalsIgnoreCase(paidBy)) {
36                 continue;
37             } else {
38                 f += memL[i] + ": -" + split.toString();
39             }
40             if (i != memL.length - 1) {
41                 f += ", ";
42             }
43         }
44         expense.setSplitBetween(f);
45         Expense e = er.save(expense);
46         return e;

```

Figure(27)- Expense Service

```

    package org.iiitb.splitwise.repositories;
    import org.iiitb.splitwise.model.User;
    import org.springframework.data.jpa.repository.JpaRepository;
    public interface UserRepository extends JpaRepository<User, String> {
        User findByUsername(String username);
        User findByEmailAndPassword(String email, String password);
    }

```

Figure(28)-User Repository

```

    package org.iiitb.splitwise.model;
    import javax.persistence.Column;
    import javax.persistence.Entity;
    import javax.persistence.GeneratedValue;
    import javax.persistence.GenerationType;
    import javax.persistence.Id;
    import javax.persistence.Table;
    import lombok.AllArgsConstructor;
    import lombok.Data;
    import lombok.NoArgsConstructor;
    @Entity
    @Table(name = "users")
    public class User {
        @Column(name = "userId", unique = true, nullable = false, updatable = false)
        @GeneratedValue(strategy = GenerationType.AUTO)
        private Integer userId;
        @Id
        @Column(name = "username", unique = true, length = 20, nullable = false)
        private String username;
        @Column(name = "password", length = 100, nullable = false)
        private String password;
        @Column(name = "name", length = 50, nullable = false)
        private String name;
        @Column(name = "email", length = 100, unique = true, nullable = false)
        private String email;
        @Column(name = "groups", length = 100000)
        private String groups;
    }

```

Figure(29)-User file

# 6. Results and Discussion

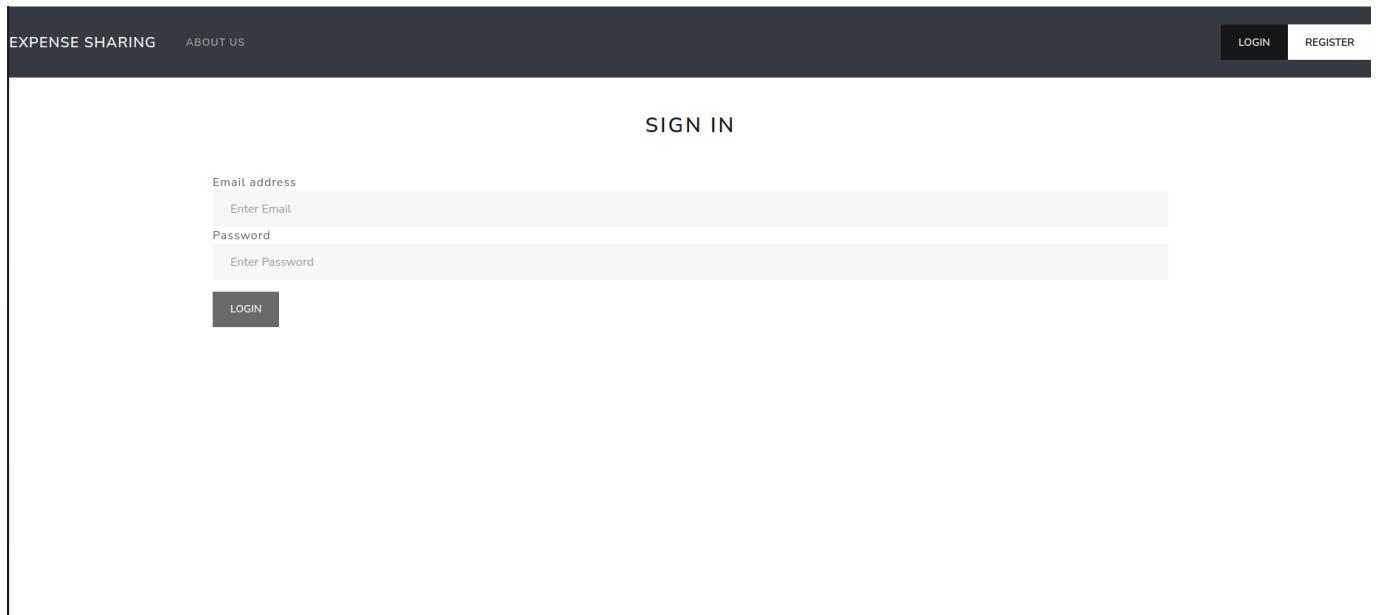
## 6.1 Home Page

The screenshot shows the homepage of a website called "EXPENSE SHARING". At the top, there is a dark navigation bar with the site name and links for "ABOUT US", "LOGIN", and "REGISTER". Below the navigation bar, a large white box contains the text "TRACK YOUR EXPENSES!" followed by a subtext: "Split bills for shared group expenses. We tell you who owes whom and how to settle debts in groups." Below this text is a cartoon illustration of seven diverse people wearing face masks, standing together. Some individuals are holding items like a tablet, a basket of fruit, and a backpack.

## 6.2 Sign Up Page

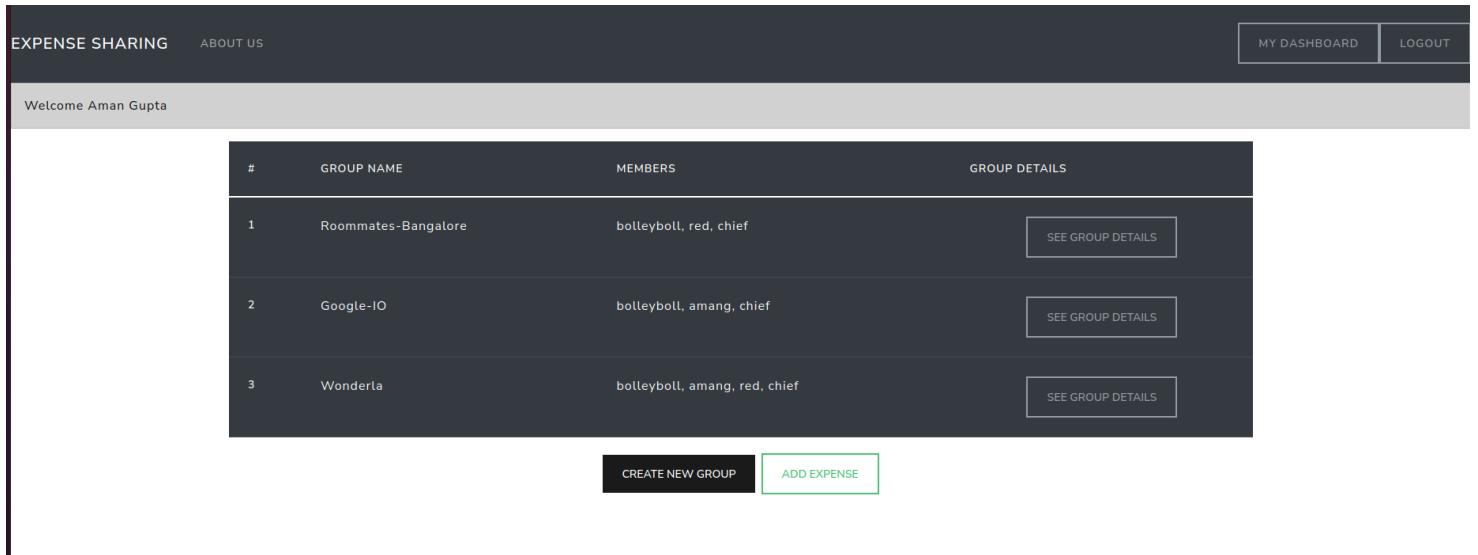
The screenshot shows the "NEW USER REGISTRATION FORM" page. It features a dark header with the site name and links for "ABOUT US", "LOGIN", and "REGISTER". The main form area has several input fields: "Username" (placeholder: "Enter your username"), "Name" (placeholder: "Enter your name"), "Email address" (placeholder: "Enter your email address"), "Password" (placeholder: "Password"), and "Confirm Password" (placeholder: "Re-enter your Password"). A "SUBMIT" button is located at the bottom left of the form.

## 6.3 Sign In Page



The screenshot shows the sign-in page of the Expense Sharing application. At the top, there is a dark header bar with the text "EXPENSE SHARING" and "ABOUT US" on the left, and "LOGIN" and "REGISTER" on the right. Below the header, the main content area has a light gray background. In the center, the word "SIGN IN" is displayed in a bold, uppercase font. Below it, there are two input fields: one for "Email address" containing the placeholder "Enter Email" and another for "Password" containing the placeholder "Enter Password". To the right of the password field is a small "Forgot Password?" link. At the bottom left of the form is a dark "LOGIN" button.

## 6.4 User Dashboard



The screenshot shows the user dashboard of the Expense Sharing application. At the top, there is a dark header bar with the text "EXPENSE SHARING" and "ABOUT US" on the left, and "MY DASHBOARD" and "LOGOUT" on the right. Below the header, the main content area has a light gray background. In the top left corner, there is a welcome message "Welcome Aman Gupta". The central part of the dashboard is a table listing three groups:

| # | GROUP NAME          | MEMBERS                      | GROUP DETAILS                     |
|---|---------------------|------------------------------|-----------------------------------|
| 1 | Roommates-Bangalore | olleyboll, red, chief        | <a href="#">SEE GROUP DETAILS</a> |
| 2 | Google-iO           | olleyboll, amang, chief      | <a href="#">SEE GROUP DETAILS</a> |
| 3 | Wonderla            | olleyboll, amang, red, chief | <a href="#">SEE GROUP DETAILS</a> |

At the bottom of the dashboard, there are two buttons: "CREATE NEW GROUP" (dark background) and "ADD EXPENSE" (light green background).

## 6.5 Group Details

### GOOGLE-IO

#### Members

- 1). bolleyboll
- 2). amang
- 3). chief

#### Expenses

The '+' before an amount denotes the Person who initially levied the costs, others are to the amount listed after their usernames.

| # | EXPENSE NAME | AMOUNT | SPLIT DETAILS                   |
|---|--------------|--------|---------------------------------|
| 1 | Cab          | 150    | amang: +150, chief: -100        |
| 2 | Sandwich     | 150    | red: -75, chief: +150           |
| 3 | Lunch        | 2500   | bolleyboll: +2500, amang: -1250 |

ADD A NEW MEMBER

## 6.6 Adding Expenses

The screenshot shows a dark-themed web application for expense sharing. At the top, there's a navigation bar with links for 'EXPENSE SHARING', 'ABOUT US', 'MY DASHBOARD', and 'LOGOUT'. Below the navigation, a form is displayed for adding a new expense. The form fields include:

- A dropdown menu labeled "For which group was the expense made" with options like "GOOGLE-IO", "Work", "Personal", and "Groceries".
- An "Expense Name" input field containing "0".
- An "Amount" input field containing "0".
- A "SUBMIT" button at the bottom of the form.

## 6.7 Adding Group

The screenshot shows a web application interface for creating a new group. At the top, there is a dark header bar with the text "EXPENSE SHARING" and "ABOUT US" on the left, and "MY DASHBOARD" and "LOGOUT" on the right. Below the header, the main content area has a light gray background. In the center, the title "NEW GROUP CREATION FORM" is displayed in bold capital letters. Below the title is a form field labeled "Group Name" with a placeholder "Enter your name". At the bottom of the form is a dark gray "SUBMIT" button.

EXPENSE SHARING ABOUT US

MY DASHBOARD LOGOUT

NEW GROUP CREATION FORM

Group Name  
Enter your name

SUBMIT

# 7. Scope for Future Work

## 1. Payment Integrations

We can make this application integrate with payment merchants to enable users to settle the bill on the application itself.

## 2. Currency Conversion

One feature that can be added in this application is for users that live in different countries. We can enable the currency to be converted based on the current currency rates in the world.

## 3. Receipt scanning

This feature will enable the users to just scan the receipt/bill and the whole information will be reflected on the application that will reduce the manual work that has to be done by the users.

## 4. Multilingual

This application can be transformed into multi-language application thus increasing the accessibility of the users from different parts of the country.

---

## 8. Conclusion

We successfully built the web application Expense Sharing Application that can be used between friends and family to split the bills easily and conveniently . This application can prove to be very useful in day to day activities.

The project taught us a great deal about all the DevOps tools that we used for our projects. Through usage of these tools we understood the importance of automation in software development. We learnt about technologies like AngularJS which was used to make the front-end of our website, SpringBoot to make the back-end, SCM tool of Git, CI/CD tools like Jenkins, Deployment platform like Docker, and CM tool like ELK. These tools diminished our work impressively.

## 9. References

1. Angular SpringBoot JWT Authentication :-

<https://loizenai.com/angular-10-spring-boot-jwt-authentication-example/#angular-spring-boot-jwt-authentication-example>

2. Angular + SpringBoot + MySQL CRUD Tutorial from “B2 Tech” :-

<https://youtube.com/playlist?list=PLA7e3zmT6XQXgBjcPyePwqgpEkI14MEXE>

3. Angular + SpringBoot CRUD Full stack Application from “Java Guides” :-

<https://youtube.com/playlist?list=PLGRDMO4rOGcNzi3CpBWscdQSzbjdWWy-f>

4. <https://www.docker.com/>

5.<https://wkrzywiec.medium.com/build-and-run-angular-application-in-a-docker-container-b65dbbc50be8>

6. <https://www.ansible.com/overview/how-ansible-works>

