

### 

### CONTINUOUS INTEGRATION AND CONTINUOUS DEPLOYMENT USING JENKINS – EVENT MANAGEMENT SYSTEM

### 

### A MINI DEVOPS PROJECT REPORT

### by

### Team – A8

### T. LAVANYA (21JG1A0560) M. NEELIMA (21JG1A0563)

B. SAHITYA (21JG1A0515) G. BHAVISHYA(21JG1A0528)

Under the esteemed guidance of

**Dr.P.V.S.Lakshmi Jagadamba**

**Professor and HOD**

CSE Department

## 

## **Department of Computer Science and Engineering**

## **GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN**

[Approved by AICTE NEW DELHI, Affiliated to JNTUK Kakinada]

[Accredited by National Board of Accreditation (NBA) for B.Tech. CSE, ECE & IT – Valid from 2019-20 to 2021-22]

Kommadi , Madhurawada, Visakhapatnam–530048

2023–2024

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## 

**CERTIFICATE**

This is to certify that the project report titled **“CONTINUOUS INTEGRATION AND CONTINUOUS DEPLOYMENT USING JENKINS-EVENT MANAGEMENT SYSTEM”** is a bonafide work of following III B.Tech. students in the Department of Computer Science and Engineering, Gayatri Vidya Parishad College of Engineering for Women affiliated to JNT University, Kakinada during the academic year 2023-2024 Semester-1.

### T. LAVANYA (21JG1A0560) M. NEELIMA (21JG1A0563)

B. SAHITYA (21JG1A0515) G. BHAVISHYA(21JG1A0528)

#### Dr.P.V.S.Lakshmi Jagadamba

#### Professor and HOD

#### CSE Department

**TABLE OF CONTENTS**

#### TOPICS PAGENO.

|  |  |  |
| --- | --- | --- |
|  | **Abstract** |  |
| **1.** | **WEB APPLICATION**  1.1 Introduction  1.2 Hardware and Software requirements  1.3 Methodology  1.4 Model architecture diagram  1.5 Implementation  1.6 Results | 1  1  2  3  4  15 |
| **2.** | **CICD USING JENKINS** |  |
|  | 2.1 Git Bash | 16 |
|  | 2.2 Jenkins |  |
|  | 2.2.1 Installation of Jenkins  2.2.2 Unlocking Jenkins | 22  25 |
|  | 2.2.3 Jenkins Plugins  2.2.4 Building a job in Jenkins  2.2.5 Implementation of Jenkins Pipelines  2.2.6 Results | 27  28  30 |
| **3.** | **CONCLUSION** | 35 |
|  |  |  |

**ABSTRACT**

In an era marked by the abundance of digital content, finding the perfect book to read can be a daunting task. This book recommendation system harnesses the capabilities of Flask, Numpy, Pandas, HTML, CSS, and Bootstrap to simplify and enhance the book discovery process. By leveraging cutting-edge data analytics and machine learning, it provides personalized book suggestions based on user preferences and historical reading patterns. The user-friendly web interface, designed with HTML, CSS, and Bootstrap, offers seamless navigation and responsiveness across various devices. This system not only transforms the way readers discover books but also serves as a valuable tool for authors, publishers, and readers' communities. Whether you're an avid reader or a developer interested in recommendation systems, this project exemplifies the potential of technology to revolutionize personalized book discovery.

**Key words:**

Book recommendation system, HTML, CSS, JavaScript, Bootstrap, GitHub, GitBash, Team Repository, Git push commands, Jenkins, Jenkins Plugins, Build, Deployment.

|  |
| --- |
|  |
|  |
|  |
|  |

**1.WEB APPLICATION**

* 1. **Introduction:**

In the digital age, information overload is a common challenge for readers seeking their next literary adventure. To address this, we present a sophisticated Book Recommendation System.This system revolutionizes the way readers discover books, making the process efficient, personalized, and visually appealing.The core of our recommendation system relies on cutting-edge data analytics and machine learning techniques. Numpy and Pandas play a pivotal role in processing and analyzing vast datasets of books, user preferences, and historical reading patterns. These tools enable us to extract meaningful insights and create accurate user profiles, resulting in highly personalized book recommendations.The user interface of our system is designed with the user in mind. HTML and CSS provide a sleek and intuitive web-based platform where readers can easily explore, search, and interact with their book recommendations. The responsive design, achieved through Bootstrap, ensures an optimal experience across various devices, from desktops to smartphones.

**Key Features:**

**Personalized Recommendations**: Our system considers a user's reading history, genre preferences, and ratings to suggest books that align with their taste.

**Search and Filter**: Users can search for specific books, authors, or genres and apply filters to refine their search results.

**User Profiles**: Readers can create and update profiles, allowing the system to continuously improve its recommendations.

This Book Recommendation System not only enhances the way readers discover books but also serves as a valuable tool for authors, publishers, and librarians by providing insights into reader preferences and trends. With its blend of advanced technology and user-centric design, our system represents a significant step forward in the world of personalized book discover

* 1. **Hardware and Software requirements:**
     1. **Hardware requirements:**
* **Server/Hosting:** You will need a server to host your Flask application. This can be a cloud-based server from providers like AWS, Google Cloud, or Heroku, or it can be a dedicated physical server if you prefer to manage it yourself. The exact specifications of the server depend on factors like the number of users and the complexity of the recommendation algorithm.
* **Memory (RAM):** The amount of RAM you need depends on the size of your dataset and the complexity of your recommendation algorithms. For a small system, 4GB to 8GB of RAM should be adequate, but larger datasets and more complex algorithms may require more.
* **Backup and Redundancy**: Consider setting up regular backups and redundancy to ensure data integrity and system availability
  + 1. **Software requirements:**
* **Python**: Flask is a Python web framework, so you'll need Python installed on your system. You can download it from the official Python website
* **Flask**: You'll need to install the Flask framework to create your web application. You can install Flask using pip, the Python package manage

Pip install flask

* **HTML**: You don't need any specific software for HTML, as it's a standard markup language. You can use a simple text editor (like Notepad on Windows, TextEdit on macOS, or any code editor like Visual Studio Code, Sublime Text, or Atom).
* **CSS:** Similarly, you can create and edit CSS files with a simple text editor or use specialized code editors to make the process more convenient.
* **Text Editors:**Visual Studio Code
* **Libraries :** Numpy,Pandas(version less than 2.0),pickle
  1. **Methodology:**

The book recommender system recommends the best books of the choice we want. The flask technique makes it dynamic website in which a web server port is allotted to the website. By clicking on the port we would get our website. The methodology includes many steps:

* **Data Collection and Preparation:**

Gather a dataset of books with relevant information like title, author, genre, and user reviews/ratings.

Preprocess the data, clean it, and prepare it for recommendation.

* **Recommendation Algorithm:**

Choose a recommendation algorithm. Common options include Collaborative Filtering, Content-Based Filtering, and Hybrid approaches.

Implement and fine-tune the algorithm using Python.

* **Flask Application Setup:**

Create a virtual environment and install Flask.

Organize your project structure, including directories for templates (HTML) and static files (CSS, images).

* **HTML and CSS Design:**

Design the user interface using HTML and CSS.

Create templates for displaying book recommendations, search results, and user profiles.

* **Flask Routes:**

Define routes in your Flask application. For instance:

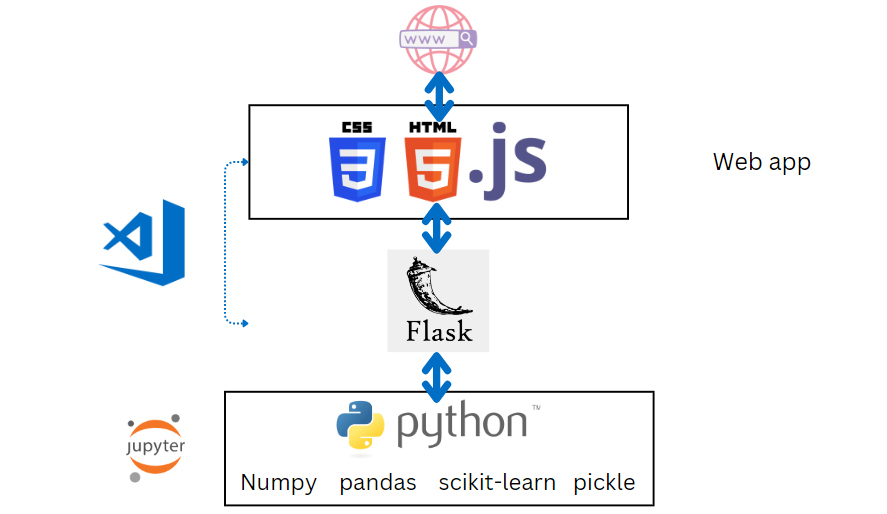
/: Home page for the recommendation system.

/recommendations: Display book recommendations.

/search: Search for specific books or authors.

/user\_profile: User profile page.

* 1. **Model architecture diagram:**



**Fig 1.4. Model Architecture diagram for Book Recommendation System**

* 1. **Implementation:**

**Source code for app.py:**

|  |
| --- |
| from flask import Flask,render\_template,request  import pickle  import numpy as np  popular\_df = pickle.load(open('popular.pkl','rb'))  pt = pickle.load(open('pt.pkl','rb'))  books = pickle.load(open('books.pkl','rb'))  similarity\_scores = pickle.load(open('similarity\_scores.pkl','rb'))  app = Flask(\_\_name\_\_)  @app.route('/')  def index():      return render\_template('index.html',                             book\_name = list(popular\_df['Book-Title'].values),                             author=list(popular\_df['Book-Author'].values),                             image=list(popular\_df['Image-URL-M'].values),                             votes=list(popular\_df['num\_ratings'].values),                             rating=list(popular\_df['avg\_rating'].values)                             )  @app.route('/recommend')  def recommend\_ui():      return render\_template('recommend.html')  @app.route('/recommend\_books',methods=['post'])  def recommend():      user\_input = request.form.get('user\_input')      index = np.where(pt.index == user\_input)[0][0]      similar\_items = sorted(list(enumerate(similarity\_scores[index])), key=lambda x: x[1], reverse=True)[1:5]      data = []      for i in similar\_items:          item = []          temp\_df = books[books['Book-Title'] == pt.index[i[0]]]          item.extend(list(temp\_df.drop\_duplicates('Book-Title')['Book-Title'].values))          item.extend(list(temp\_df.drop\_duplicates('Book-Title')['Book-Author'].values))          item.extend(list(temp\_df.drop\_duplicates('Book-Title')['Image-URL-M'].values))          data.append(item)      print(data)      return render\_template('recommend.html',data=data)  if \_\_name\_\_ == '\_\_main\_\_':      app.run(debug=True) |

Index.html source code:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8" />  <title>ReadWise - Book Recommender</title>  <link  rel="stylesheet"  href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"  integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"  crossorigin="anonymous"  />  <style>  body {  background-color: black;  }  .navbar {  background-color: #ffcc00; /\* Yellow background \*/  border-radius: 0;  }  .navbar-brand,  .nav > li > a {  color: black; /\* Solid black text color \*/  font-size: 20px; /\* Increase font size \*/  font-weight: bold; /\* Font thickness \*/  font-family: "Arial", sans-serif; /\* Choose a suitable font family \*/  transition: color 0.3s;  }  .navbar-brand:hover,  .nav > li > a:hover {  color: #007bff; /\* Blue color on hover \*/  }  .navbar-right {  margin-right: 20px;  }  .navbar-inverse .navbar-toggle .icon-bar {  color: black;  background-color: #333; /\* Color of the menu icon bars \*/  }  /\* Custom styling \*/  .center-content {  text-align: center;  margin-top: 20vh; /\* Adjust as needed for vertical centering \*/  }  .center-content h1 {  font-size: 48px;  font-weight: bold;  color: white; /\* White text color \*/  margin-bottom: 20px;  }  .center-content p {  font-size: 24px;  color: white; /\* White text color \*/  }  .center-content a {  margin-top: 30px; /\* Adjust as needed \*/  animation: bounce 2s infinite; /\* Animation effect \*/  }  .center-content a.btn-primary {  background-color: #ffcc00;  color: black;  border-color: #ffcc00;  }  .center-content a.btn-primary:hover {  background-color: #00aa00; /\* Green color on hover \*/  border-color: #00aa00;  color: white;  }  @keyframes bounce {  0%,  20%,  50%,  80%,  100% {  transform: translateY(0);  }  40% {  transform: translateY(-20px);  }  60% {  transform: translateY(-10px);  }  }  </style>  </head>  <body>  <nav class="navbar navbar-inverse">  <div class="container-fluid">  <div class="navbar-header">  <a class="navbar-brand">ReadWise</a>  </div>  <ul class="nav navbar-nav navbar-right">  <li><a href="#">Home</a></li>  <li><a href="#">Recommended</a></li>  <li><a href="#">Contact</a></li>  </ul>  </div>  </nav>  <div class="container">  <div class="row">  <div class="col-md-12 center-content">  <h1>Welcome to Book Recommender</h1>  <p>Discover books tailored just for you!</p>  <a href="/recommend" class="btn btn-primary btn-lg"  >Get Recommendations</a  >  </div>  </div>  </div>  </body>  </html> |

Recommendation.html source code:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8" />  <title>Book Recommender System</title>  <!-- Latest compiled and minified CSS -->  <link  rel="stylesheet"  href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"  integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"  crossorigin="anonymous"  />  </head>  <style>  body {  background-color: black;  }  .navbar {  background-color: #ffcc00; /\* Yellow background \*/  border-radius: 0;  }  .navbar-brand,  .nav > li > a {  color: black; /\* Solid black text color \*/  font-size: 20px; /\* Increase font size \*/  font-weight: bold; /\* Font thickness \*/  font-family: "Arial", sans-serif; /\* Choose a suitable font family \*/  transition: color 0.3s;  }  .navbar-brand:hover,  .nav > li > a:hover {  color: #007bff; /\* Blue color on hover \*/  }  .navbar-right {  margin-right: 20px;  }  .navbar-inverse .navbar-toggle .icon-bar {  color: black;  background-color: #333; /\* Color of the menu icon bars \*/  }  /\* Custom styling \*/  .center-content {  text-align: center;  margin-top: 20vh; /\* Adjust as needed for vertical centering \*/  }  .center-content h1 {  font-size: 48px;  font-weight: bold;  color: white; /\* White text color \*/  margin-bottom: 20px;  }  .center-content p {  font-size: 24px;  color: white; /\* White text color \*/  }  .center-content a {  margin-top: 30px; /\* Adjust as needed \*/  animation: bounce 2s infinite; /\* Animation effect \*/  }  .center-content a.btn-primary {  background-color: #ffcc00;  color: black;  border-color: #ffcc00;  }  .center-content a.btn-primary:hover {  background-color: #00aa00; /\* Green color on hover \*/  border-color: #00aa00;  color: white;  }  .centered-content {  text-align: center;  color: white;  }  @keyframes bounce {  0%,  20%,  50%,  80%,  100% {  transform: translateY(0);  }  40% {  transform: translateY(-20px);  }  60% {  transform: translateY(-10px);  }  }  .card-container {  margin-top: 50px;  padding: 20px;  border: 1px solid white;  border-radius: 10px;  background-color: rgba(255, 255, 255, 0.1);  text-align: center;  box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);  transition: background-color 0.3s;  }  .card-container:hover {  background-color: rgba(255, 255, 255, 0.2);  }  .card-container img {  max-width: 100%;  border-radius: 5px;  margin-bottom: 10px; /\* Add space below the image \*/  }  .card-container p {  color: white;  font-size: 16px; /\* Adjust the font size as needed \*/  margin: 10px 0; /\* Add space above and below the paragraph \*/  }  .btn-recommend {  background-color: white;  color: black;  border-color: white;  transition: background-color 0.3s, color 0.3s;  }  .btn-recommend:hover {  background-color: #ffcc00;  color: black;  }  </style>  <body style="background-color: black">  <nav class="navbar navbar-inverse">  <div class="container-fluid">  <div class="navbar-header">  <a class="navbar-brand">My Book Recommender</a>  </div>  <ul class="nav navbar-nav navbar-right">  <li><a href="#">Home</a></li>  <li><a href="/recommend">Recommended</a></li>  <li><a href="#">Contact</a></li>  </ul>  </div>  </nav>  <div class="container">  <div class="row">  <div class="col-md-12">  <div class="centered-content">  <h1 class="text-white" style="font-size: 50px">Recommend Books</h1>  <form action="/recommend\_books" method="post">  <input  name="user\_input"  type="text"  class="form-control text-center"  /><br />  <button  type="submit"  class="btn btn-lg btn-recommend text-center"  >  Get Recommendations  </button>  </form>  </div>  </div>  {% if data %} {% for i in data %}  <div class="col-md-3">  <div class="card-container">  <img class="card-img-top" src="{{i[2]}}" />  <p>{{i[0]}}</p>  <h4>{{i[1]}}</h4>  </div>  </div>  {% endfor %} {% endif %}  </div>  </div>  </body>  </html> |

Style.css source code:

|  |
| --- |
| /\* Reset some default margins and paddings \*/  body,  h1,  h2,  h3,  h4,  h5,  h6,  p {  margin: 0;  padding: 0;  }  color on hover \*/  }  /\* Navigation items on left \*/  .navbar-nav.left {  margin-right: auto;  margin-left: 20px;  }  /\* Navigation items on right \*/  .navbar-nav.right {  margin-left: auto;  margin-right: 20px;  }  /\* Navigation bar brand (logo) \*/  .navbar-brand {  color: #333; /\* Dark text color \*/  font-size: 24px;  margin-left: 20px;  transition: color 0.3s; /\* Smooth color transition on hover \*/  }  .navbar-brand:hover {  color: #007bff; /\* Blue color on hover \*/  }  /\* Navigation bar brand on left \*/  .navbar-brand.left {  margin-right: auto;  margin-left: 20px;  }  /\* Navigation bar brand on right \*/  .navbar-brand.right {  margin-left: auto;  margin-right: 20px;  }  .container {  margin-top: 20px;  }  .container h1 {  font-size: 50px;  }  .container p {  font-size: 18px;  margin-bottom: 20px;  }  transition: background-color 0.3s; /\* Smooth color transition on hover \*/  }  .btn-primary:hover {  background-color: #0056b3;  } |

* 1. **Results:**

After clicking on the link we get the webpage as follows:

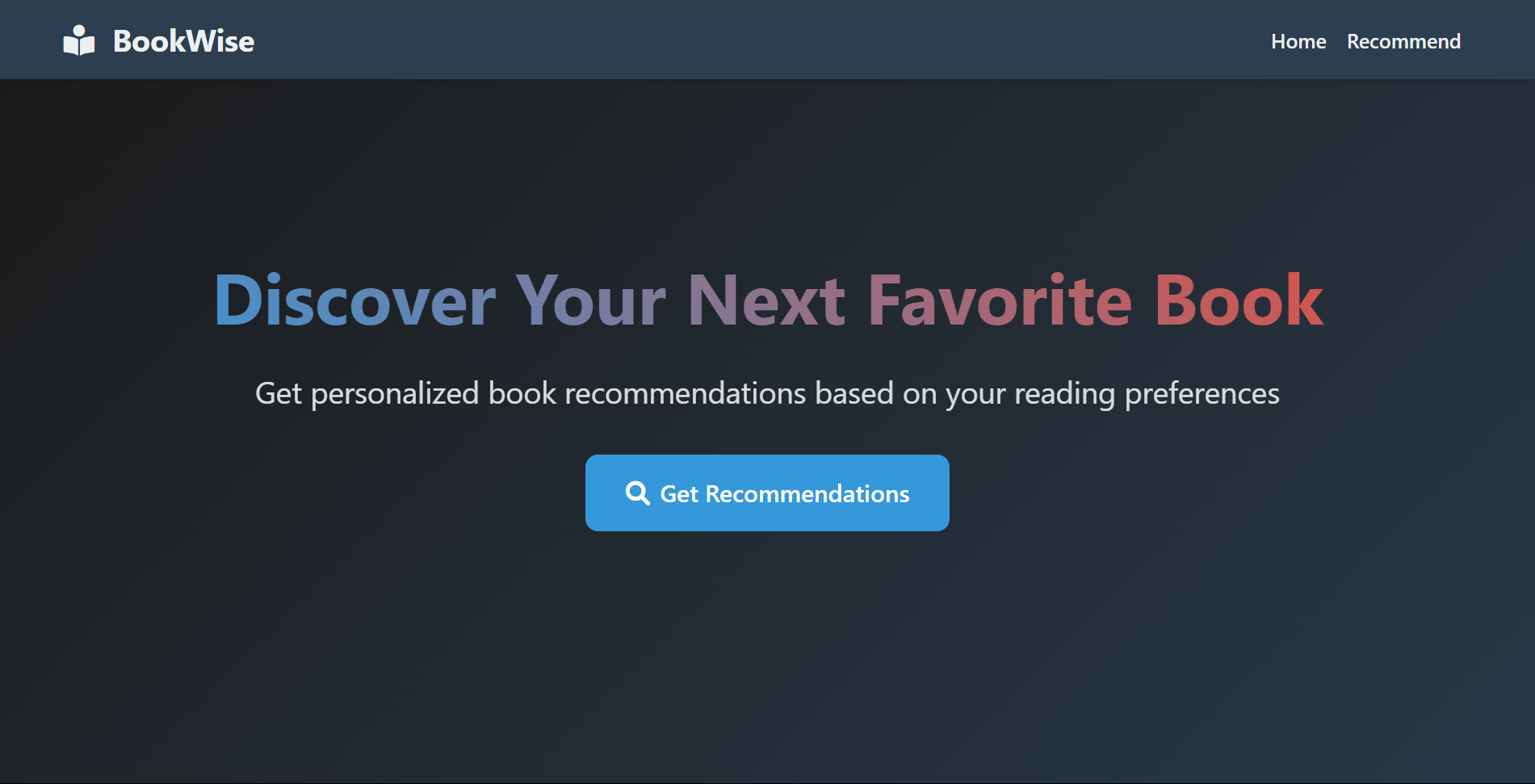
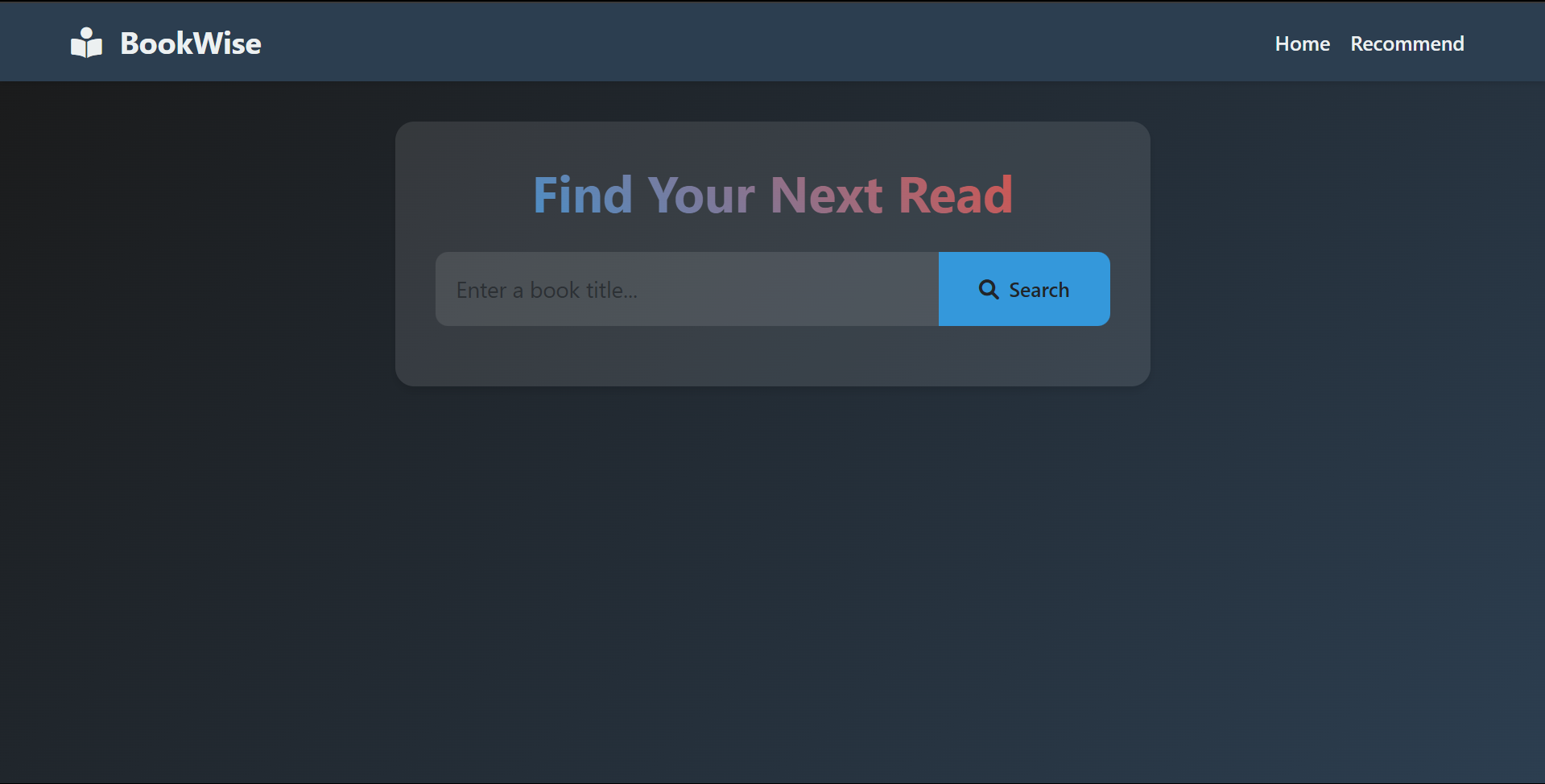


Fig 1.6.1. Homepage of website



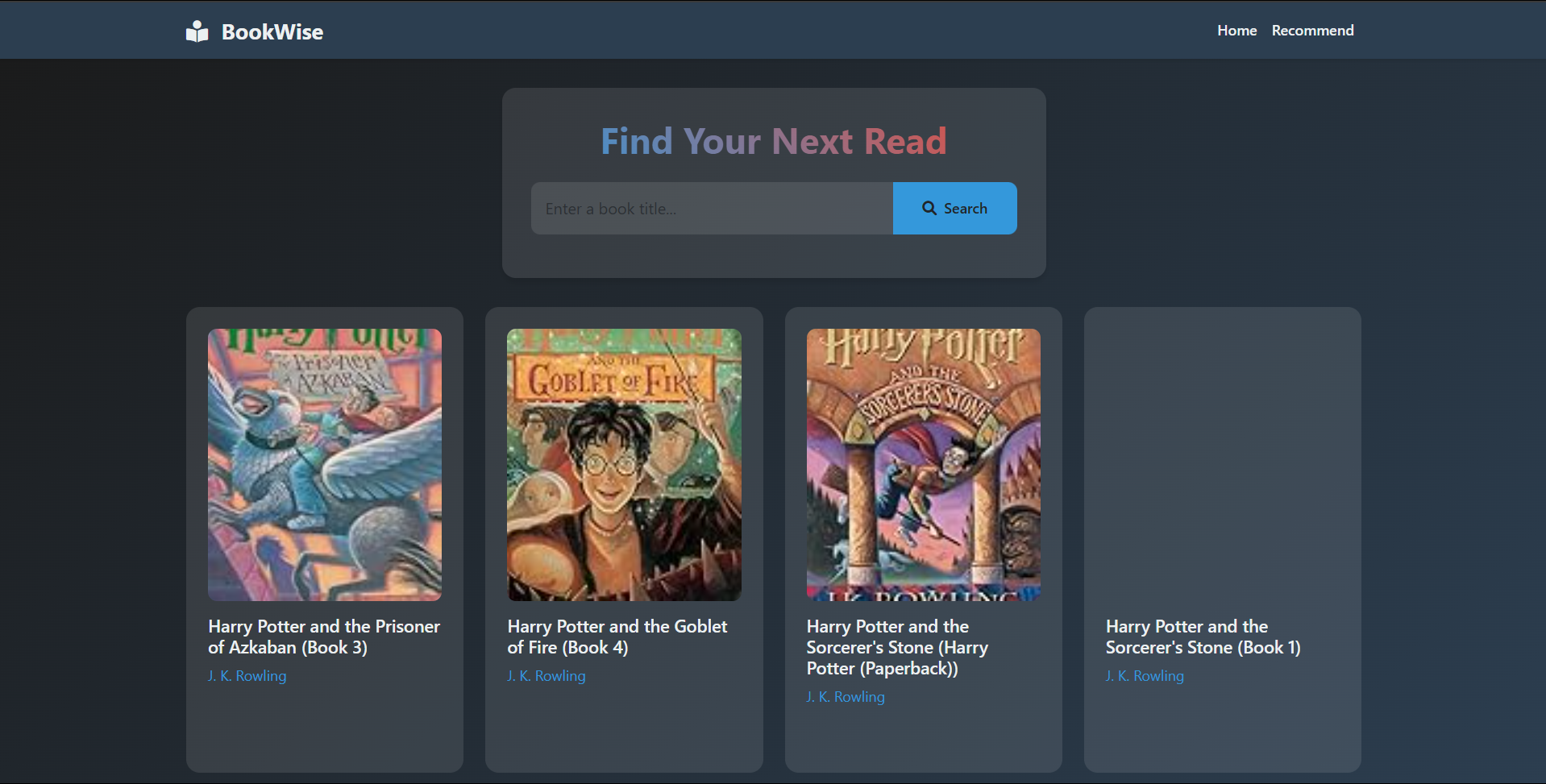


Fig 1.6.2. Get recommendation page of website

**2.** **CONTINUOUS INTEGRATION AND CONTINUOUS DEPLOYMENT USING JENKINS**

**2.1 Git Bash:**

Git Bash is an application for Microsoft Windows environments which provides an emulation layer for a Git command line experience. Bash is an acronym for Bourne Again Shell. A shell is a terminal application used to interface with an operating system through written commands.

**2.1.1 Installation and Setup of Git:**

1. Download GIT from the website below based on your system’s operating system and configuration.



Fig 2.1.1. Git search in browser.

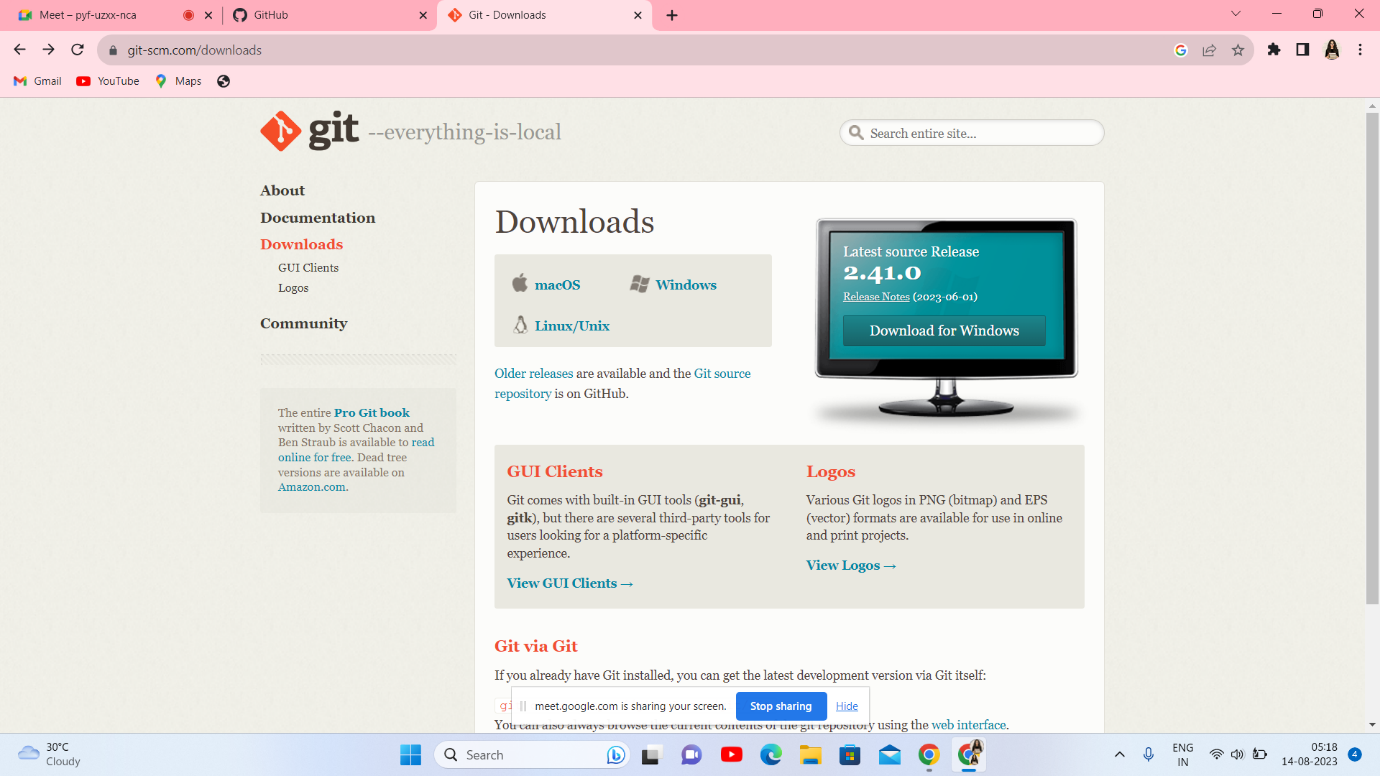


Fig 2.1.2. Git website to download the application.

1. Install GIT by clicking next for a few times.
2. After installation is complete go to start menu and search for Git Bash and open it.
3. Check the Git version using the command git –version.

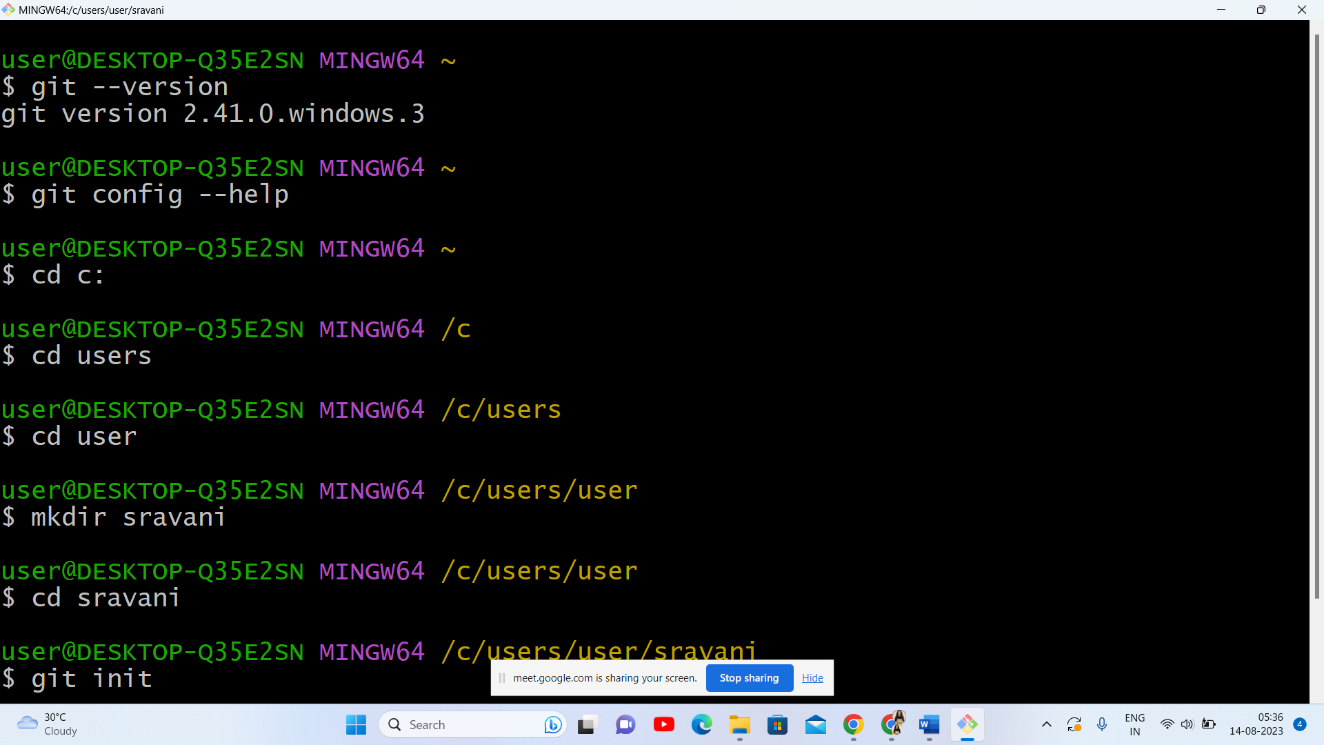


Fig 2.1.3. git version checking.

5) Creating a Git Repository:

1. Initialize a Git repository using `git init` in your local project directory.
2. Create a new repository on a GitHub.
3. Link the local repository to the remote repository using `git remote add origin <repository\_url>`.
4. Stage and commit your initial project files.
5. Push the commits to the remote repository using `git push -u origin main`.



Fig 2.1.4. Repository creation

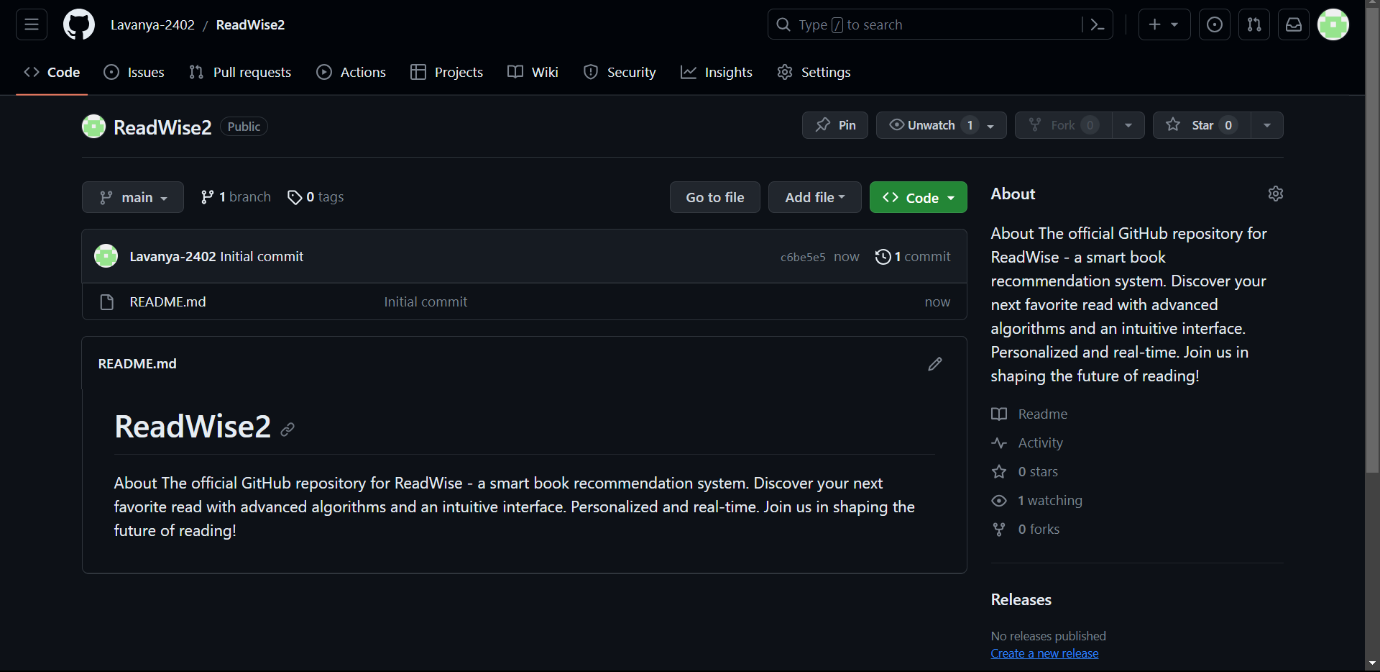


Fig 2.1.5. Git repository page.

6) Adding Collaborators:

1. Go to the GitHub.
2. Access the repository's settings and navigate to the "Collaborators" or "Access" section.
3. Invite collaborators by username granting them access to the repository.
4. Collaborators accept the invitation and clone the repository to their local machines.

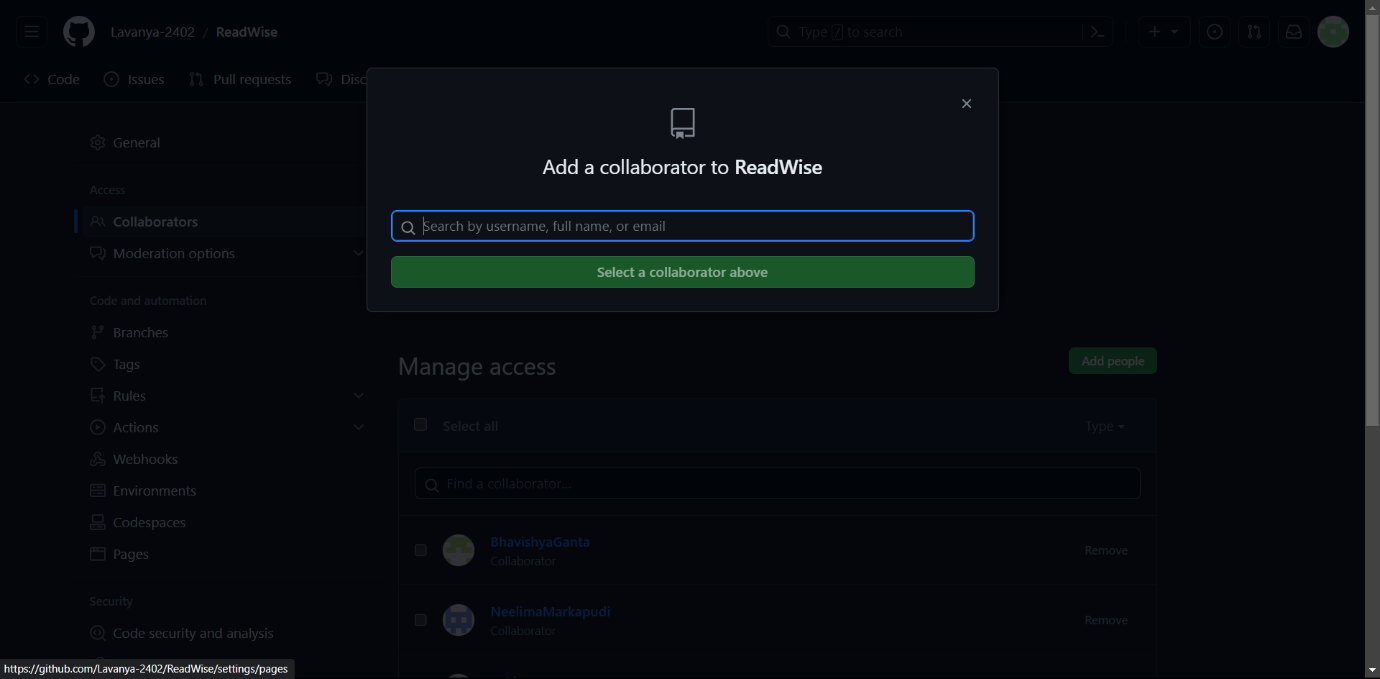


Fig 2.1.6. Searching for collaborator.

Fig2.1.7. After adding collaborators.

7) Creating Branches:

1. Create a new branch using `git branch <branch\_name>` to start working on a specific task.
2. Switch to the new branch using `git checkout <branch\_name>` or combine the branch creation and switch with `git checkout -b <branch\_name>`.
3. Develop and make changes in the branch.
4. Commit your changes on the branch using `git add` and `git commit`.

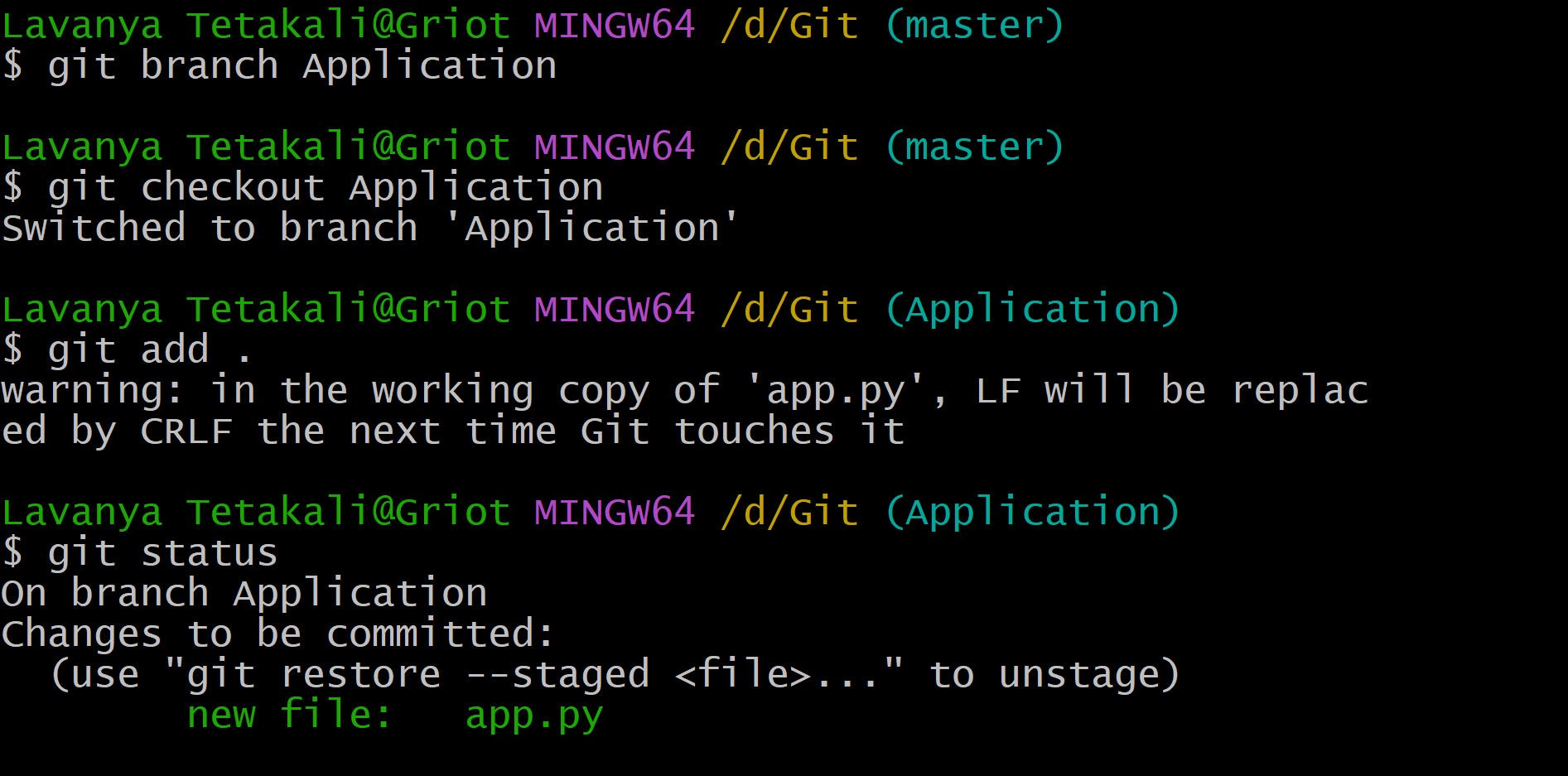


Fig 2.1.8. Branch creation and staging files into git repository.

8. Pushing Files to the Repository:

1. Push the commits from your local branch to the remote repository using `git push origin <branch\_name>`.

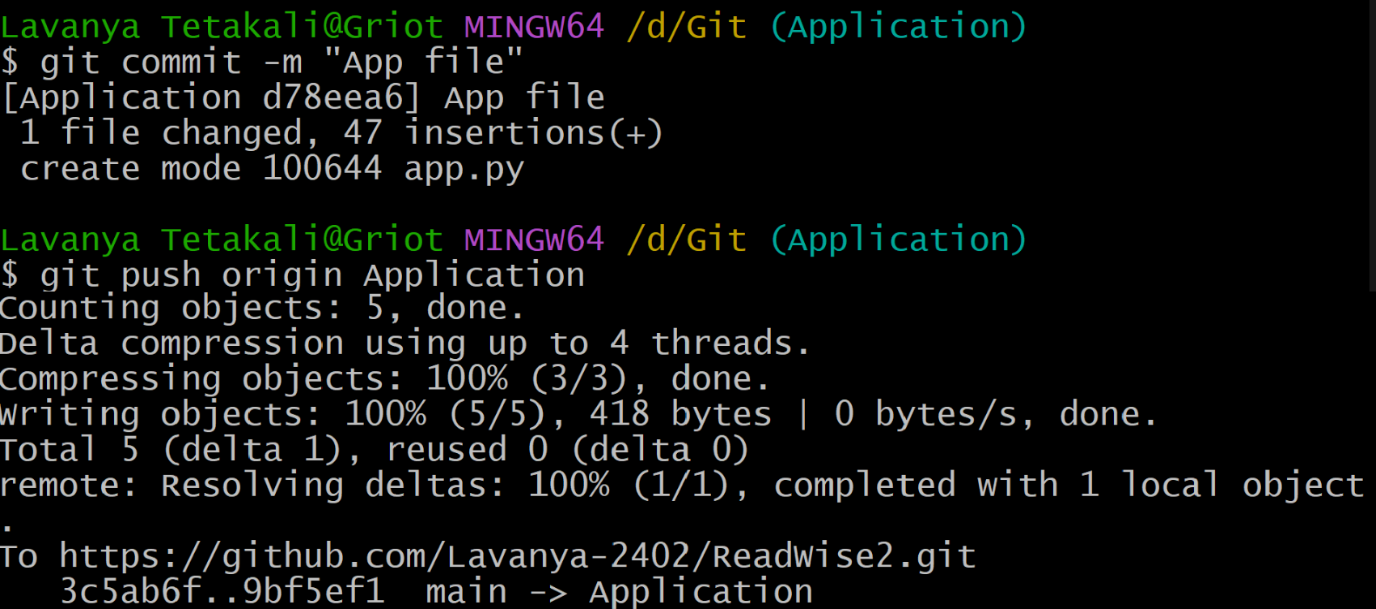


Fig 2.1.9. commiting and pushing files into git repository.

This will make your changes available for collaboration with others.

9. Merging Changes:

1. Open a Pull Request (PR) on the github to merge the feature branch into the main branch.
2. Collaborators review and discuss the code in the PR.
3. Once the PR is approved, merge the changes into the main branch.
4. Continuously update the local repository by pulling in changes from the remote repository.
5. Collaborators should keep their local branches up-to-date by pulling the latest changes from the main branch.

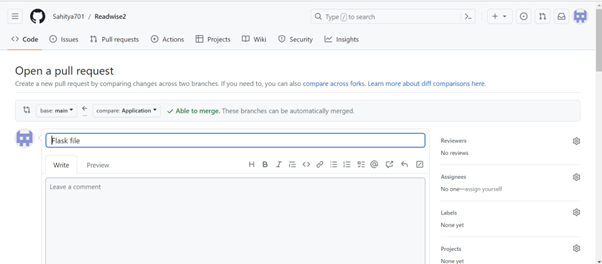


Fig 2.1.10. Pull request in github.

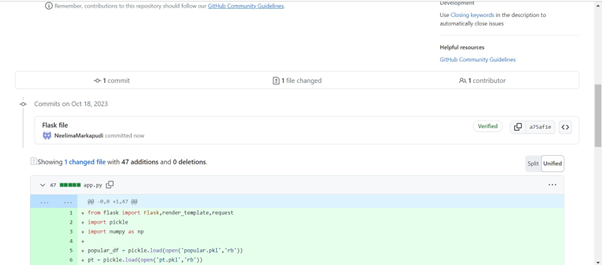


Fig 2.1.11. Pull request in github.

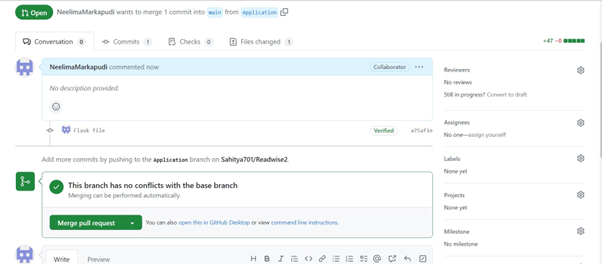


Fig 2.1.12. Merge the pull request in github.

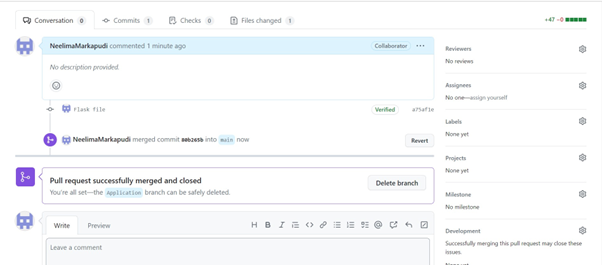


Fig 2.1.13. After merging the pull request in github.

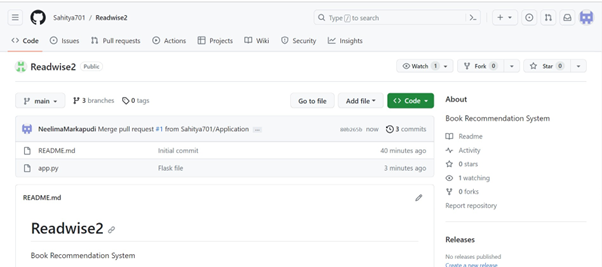


Fig 2.1.14. Main branch after merging pull request in github.

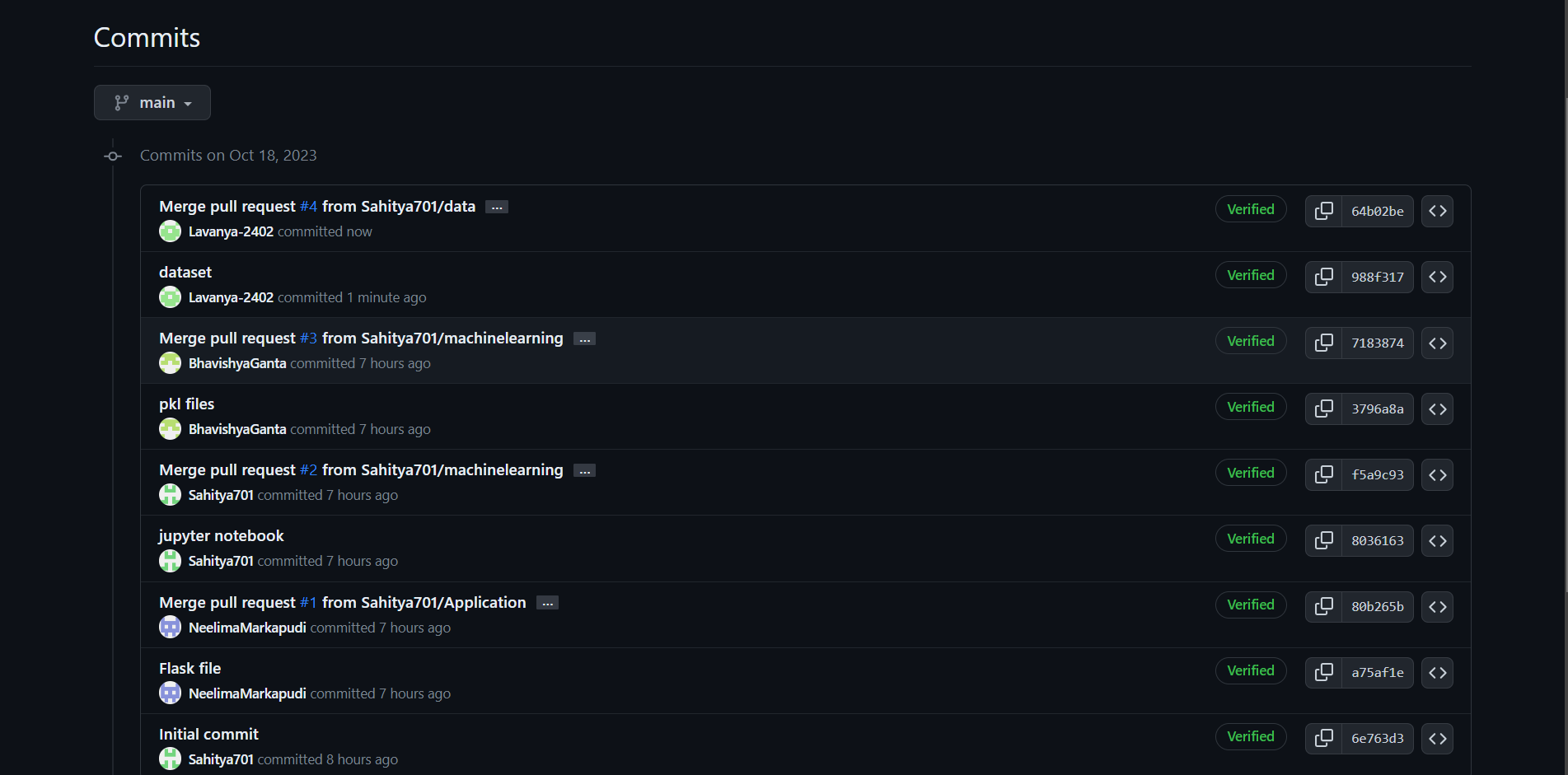


Fig 2.1.15. All the commits done by collaborators in github.

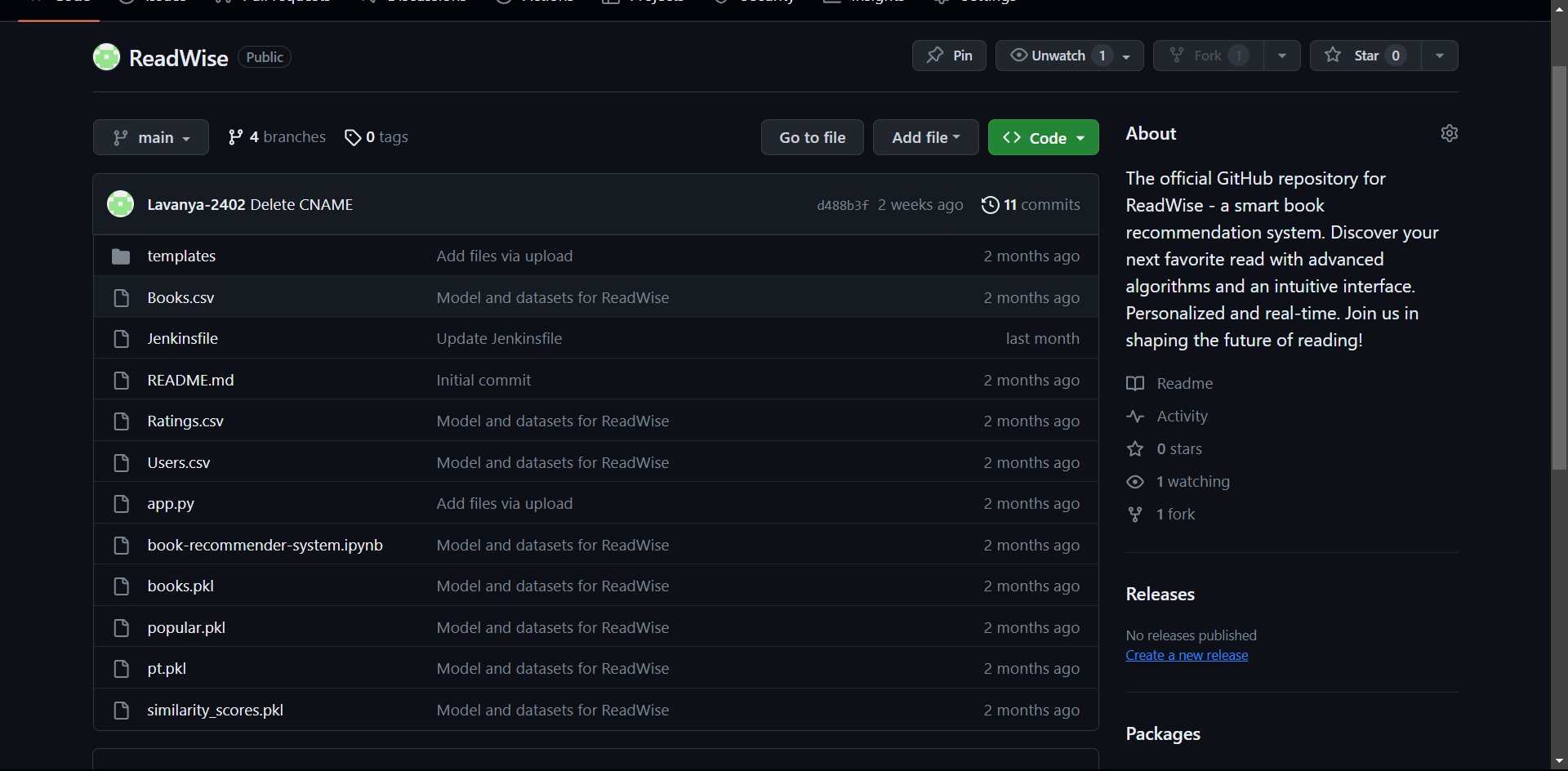


Fig 2.1.16. Github repository after completion of project.

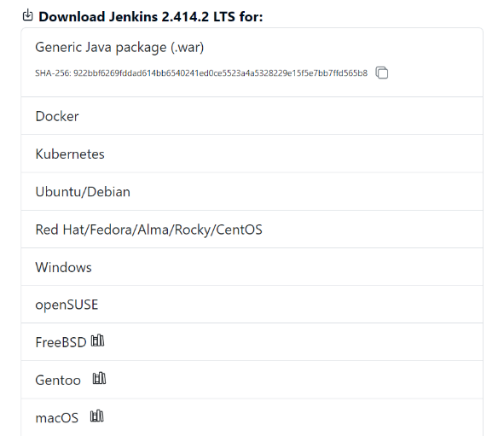
**2.2 Jenkins:**

There are 2 steps involved in setting up Jenkins completely. They are,

* Download and installation of Jenkins software
* Unlocking Jenkins with localhost

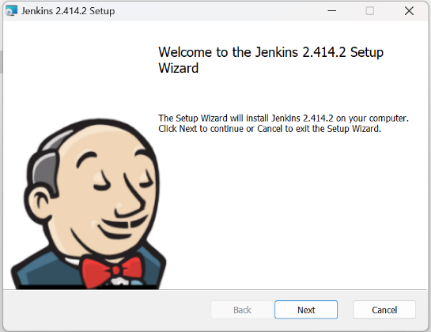
**2.2.1 Steps to download and install Jenkins:**

1. Go to the website [**https://www.jenkins.io/download/**](https://www.jenkins.io/download/)where we can get Jenkins software for any type of Operating system (Windows, Linux, Mac, etc..). Select windows for Jenkins 2.414.2 LTS.

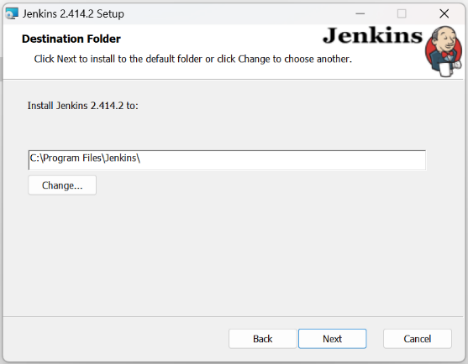
****

1. After selecting windows, Jenkins.exe file starts downloading.

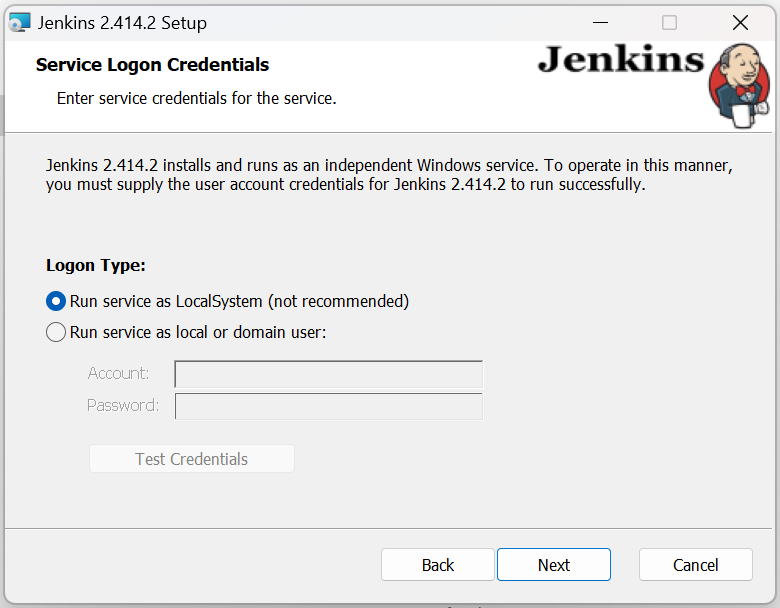
Go to Downloads -> Double click on Jenkins.exe file. Click on Next.



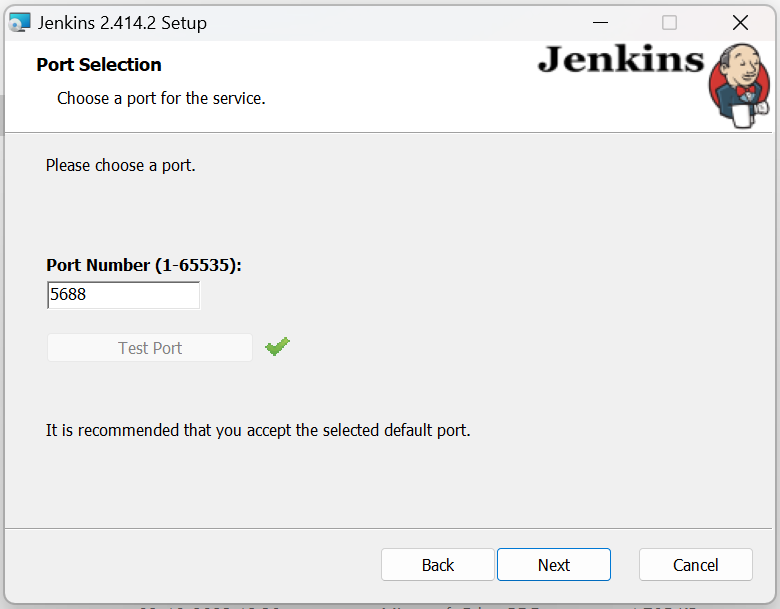
1. Now you will be asked to set the destination for the folder. There is no need to change the path. Click on Next.

****

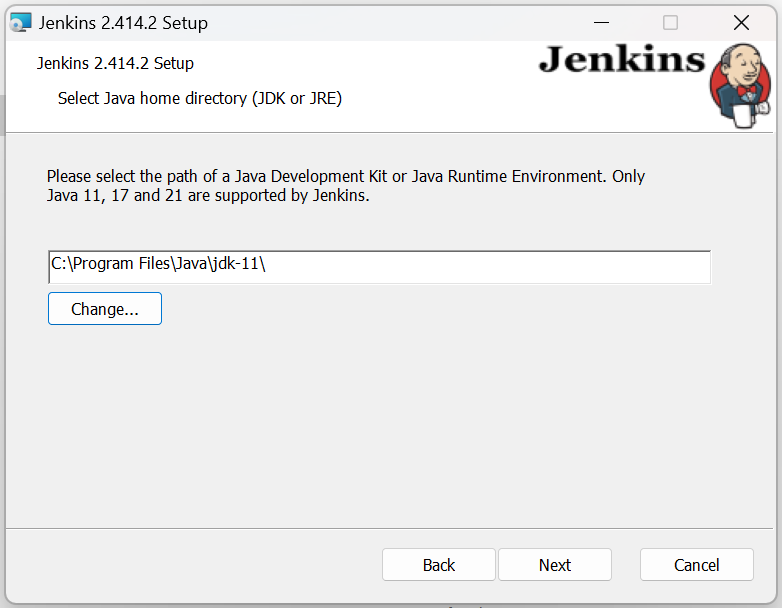
1. Now you will be asked to enter the services logon credentials. Select Run service as LocalSystem and click on Next.

****

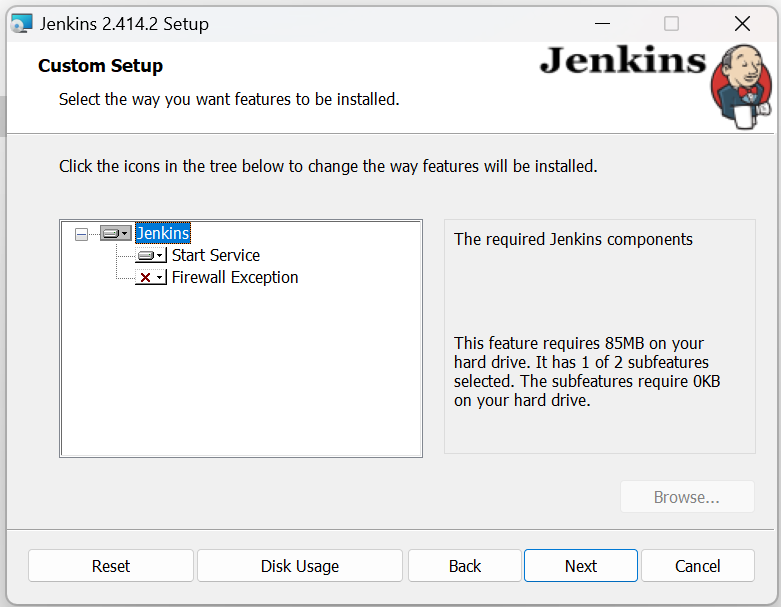
1. Now you will be asked to select a port number for localhost. By default it is 8080 if it is not available, then you have to choose one from the available ports in your device. You can use that number with localhost when the test post result is success. Click on Next.

****

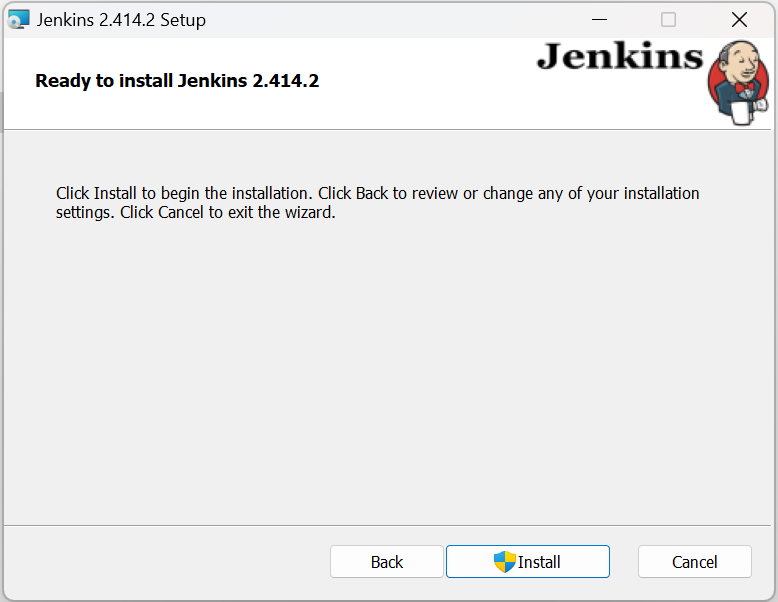
1. As Jenkins runs on Java Environment, we must give Java path to Jenkins. Jenkins supports only Java 11, 17 or 21. After giving path of Java, Click on Next.

****

1. Now a custom setup window opens. Click on Next without making any changes in it.

****

1. Click on Install. Now the installation of Jenkins begins.

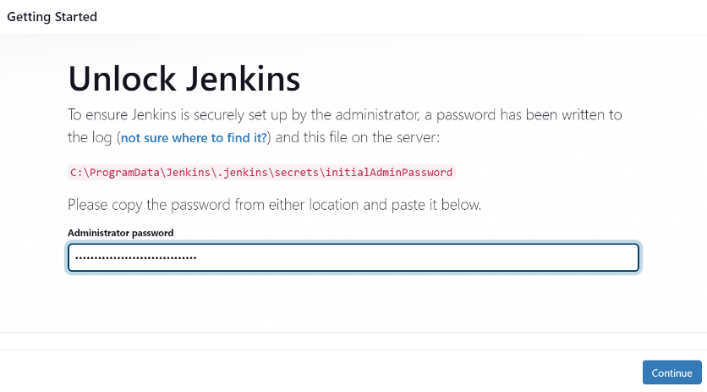
****

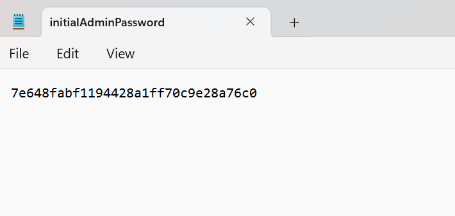
1. After the completion of Installation, Click on Finish.

****

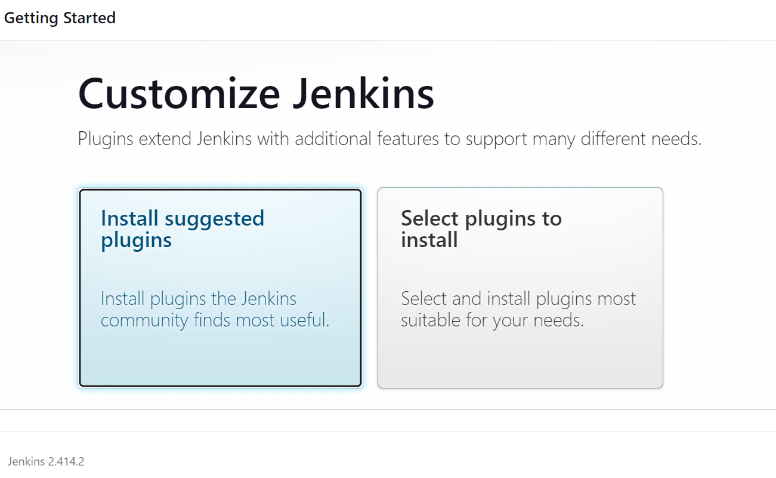
**2.2.2 Steps to unlock the Jenkins after Installation:**

1. Open your web browser. Type localhost:5688 (your port number). After pressing Enter you will be opened with a Getting started window asking for a password. To find the password open the location mentioned and copy paste the code. Click on Continue.

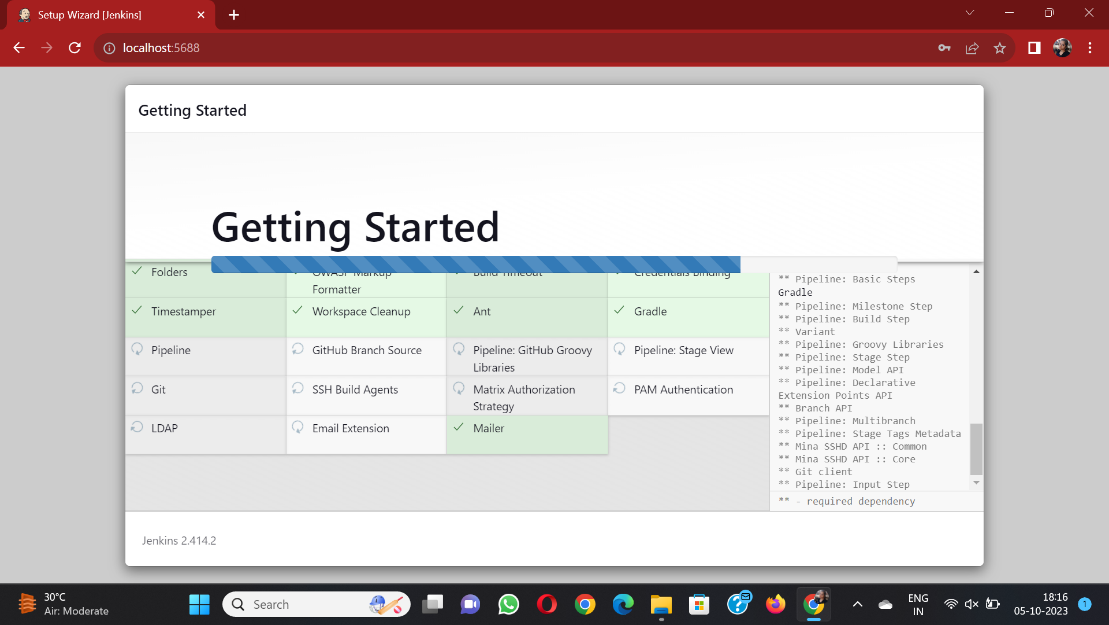
****

****

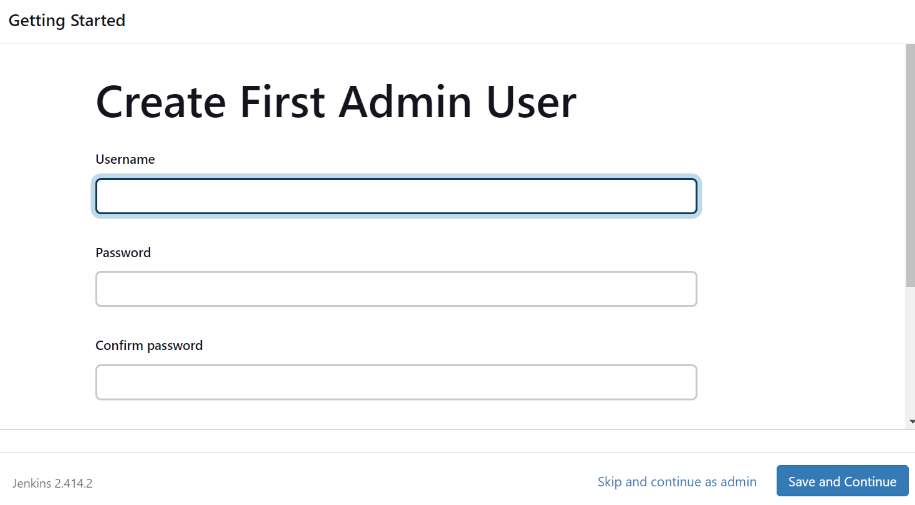
1. Select Install suggested plugins.

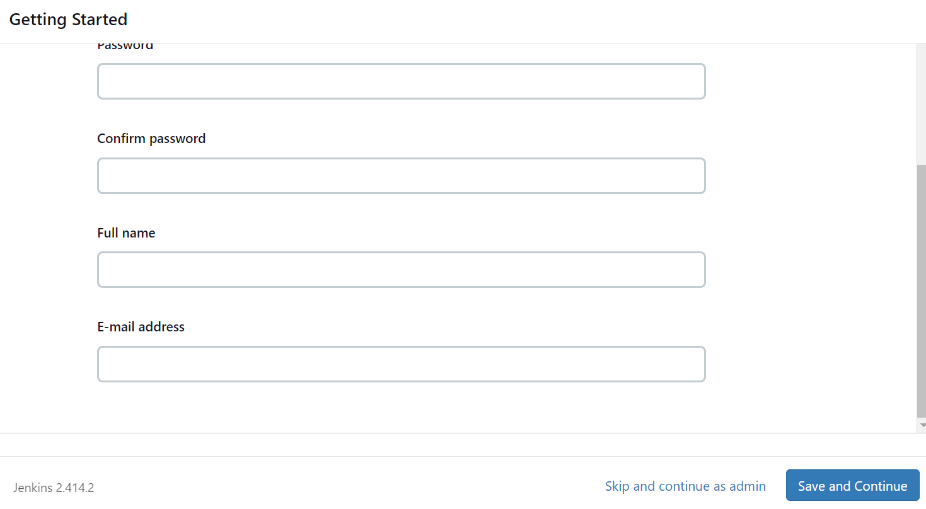
****

1. Now all the required plugins will be downloaded one by one.

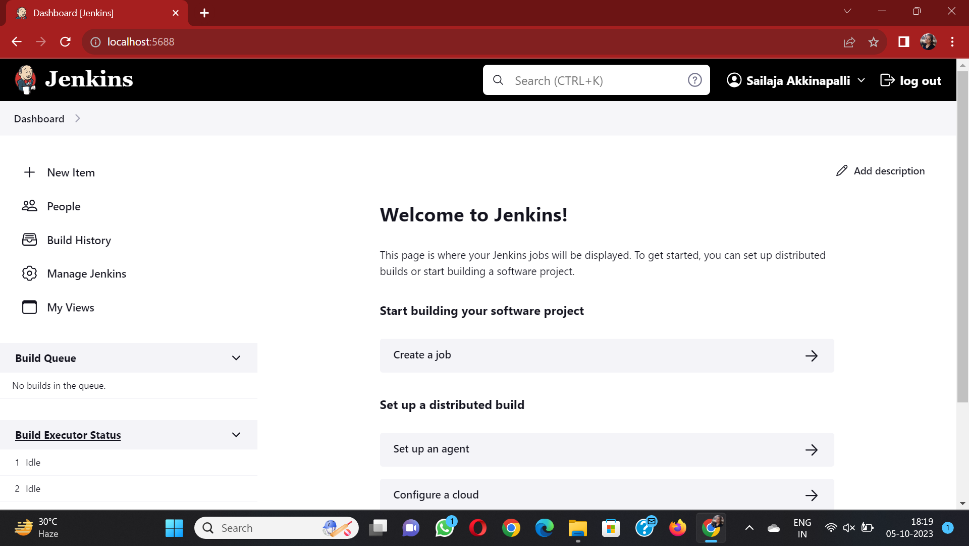
****

1. Now create an account by filling all the required details. Click on Save and Continue after filling all the details.

****

****

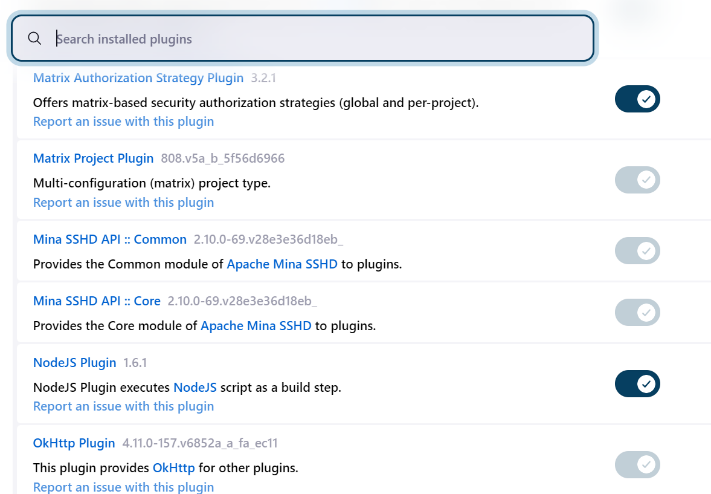
1. After clicking on Save and Continue, you successfully completed setting up your profile in Jenkins.

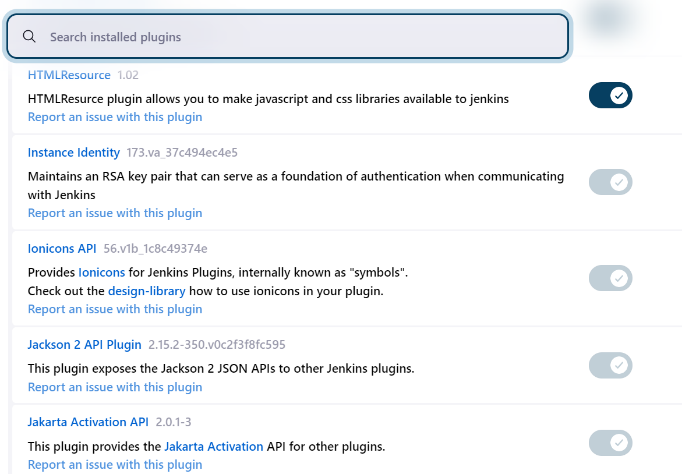
****

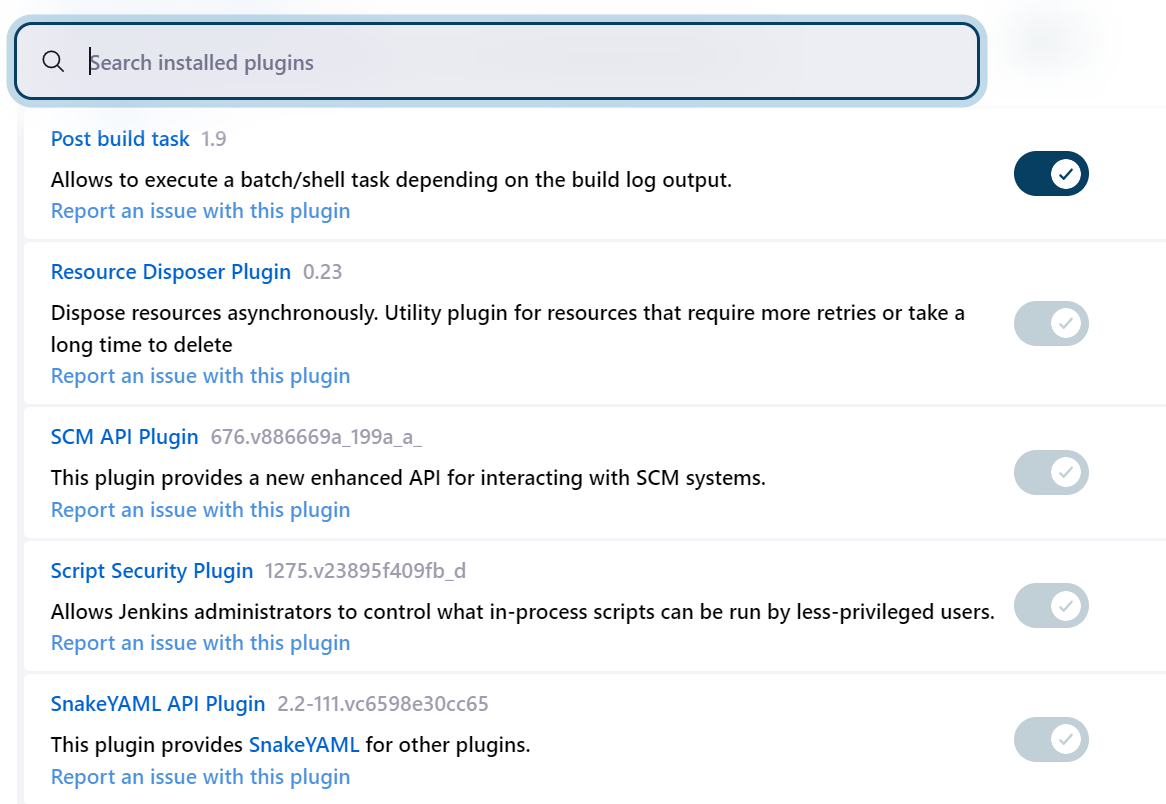
**2.2.3 Installation of additional required plugins in Jenkins:**

To make Jenkins deploy your project correctly, you should install some more additional plugins. Here is the list of plugins we installed to deploy our project and view the output.

* HTML Publisher plugin
* Subversion Workspace Cleaner
* HTMLResource
* Post build task







**2.2.4 Building a job in Jenkins:**

* Open Jenkins Dashboard. Click on New Item.
* Enter a name for Job. Select Freestyle project and Click OK.



Fig 2.2.5.1 Create a freestyle project.

* Scroll down until you reach Source Code Management. Select Git.
* Now go to your GitHub account and copy the URL of your Repository in which your project is present.
* Paste the copied URL in Repository URL section.

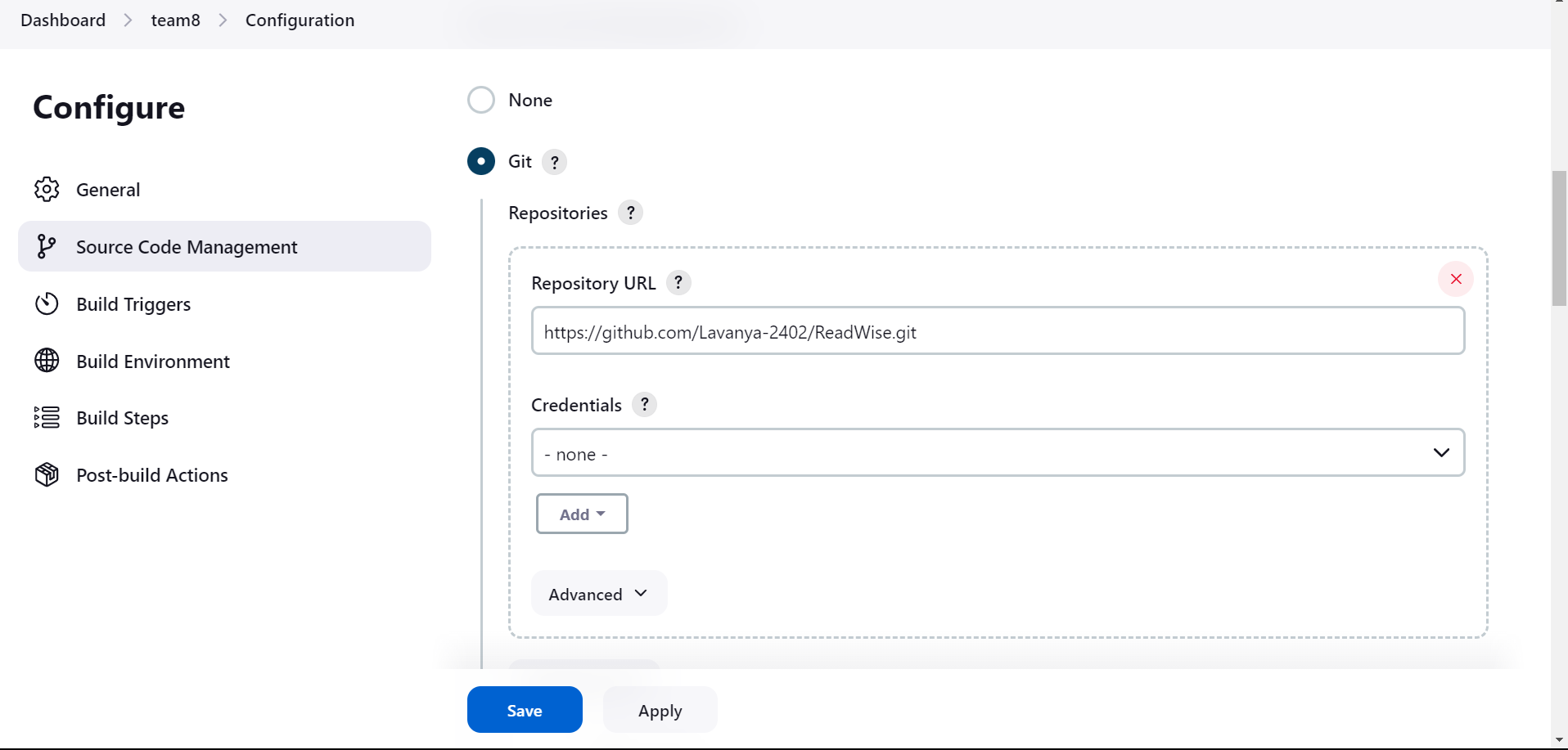


Fig 2.2.5.2 Give github URL

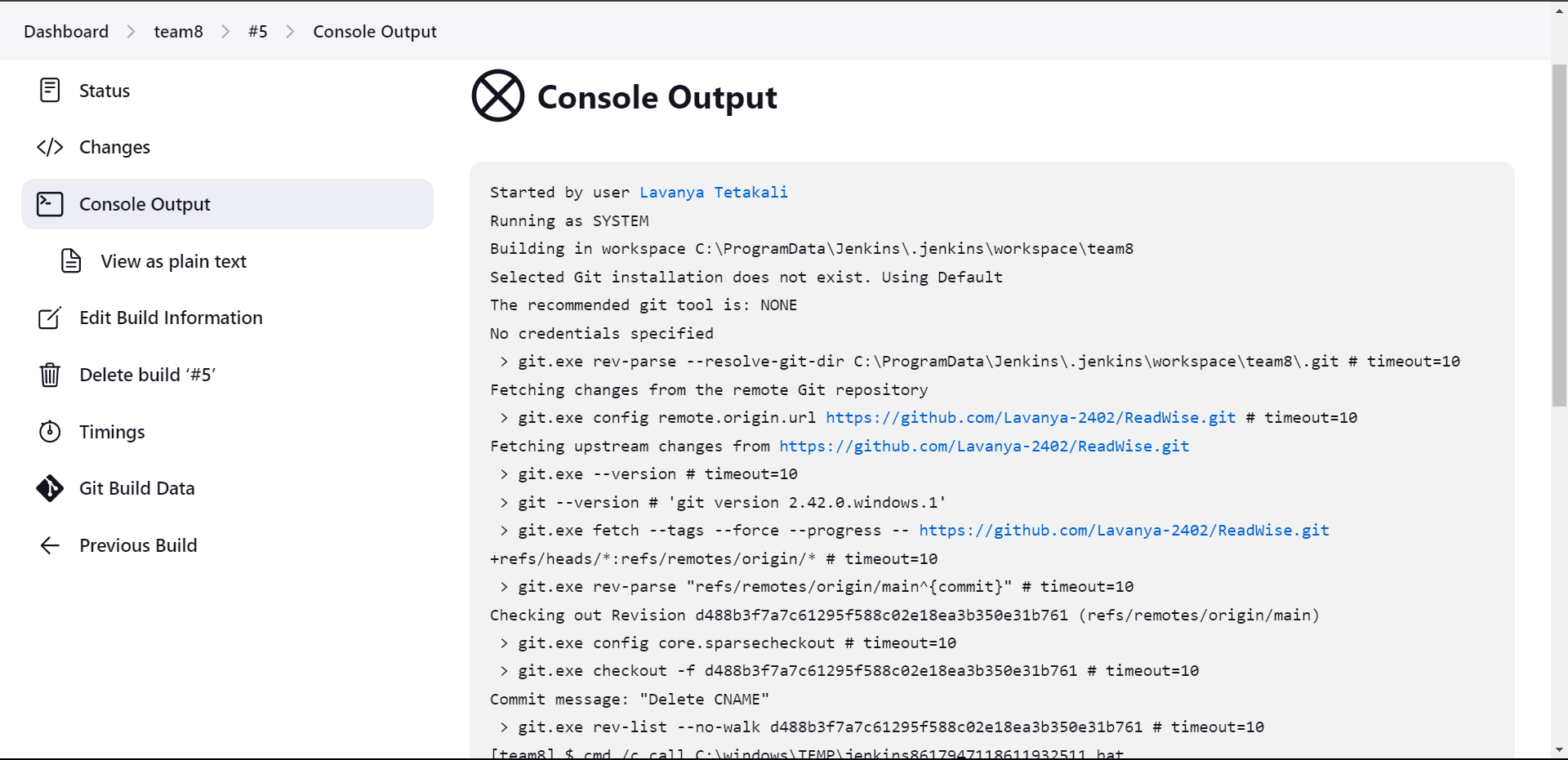


Fig 2.2.5.3 view console output

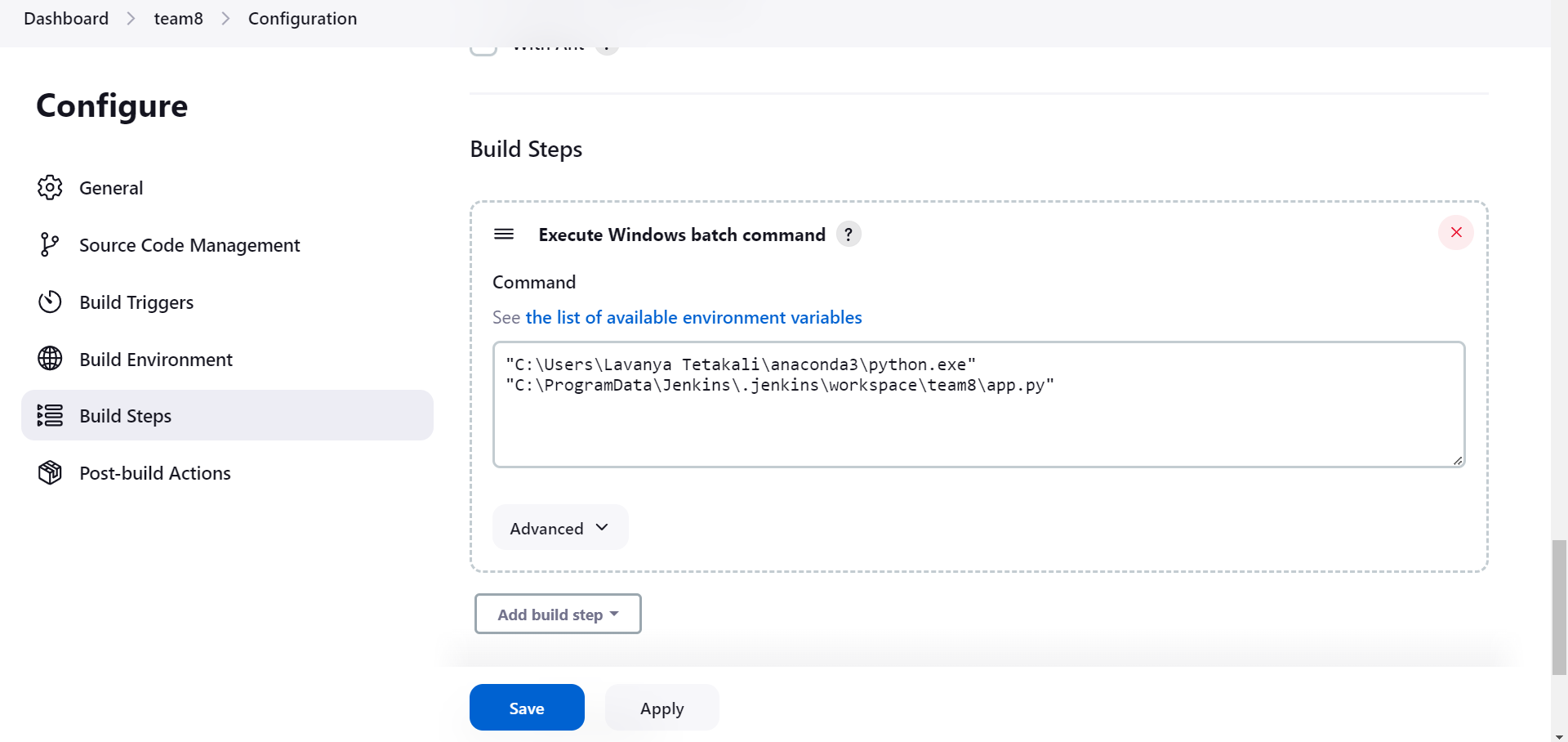


Fig 2.2.5.4 Build steps for the application

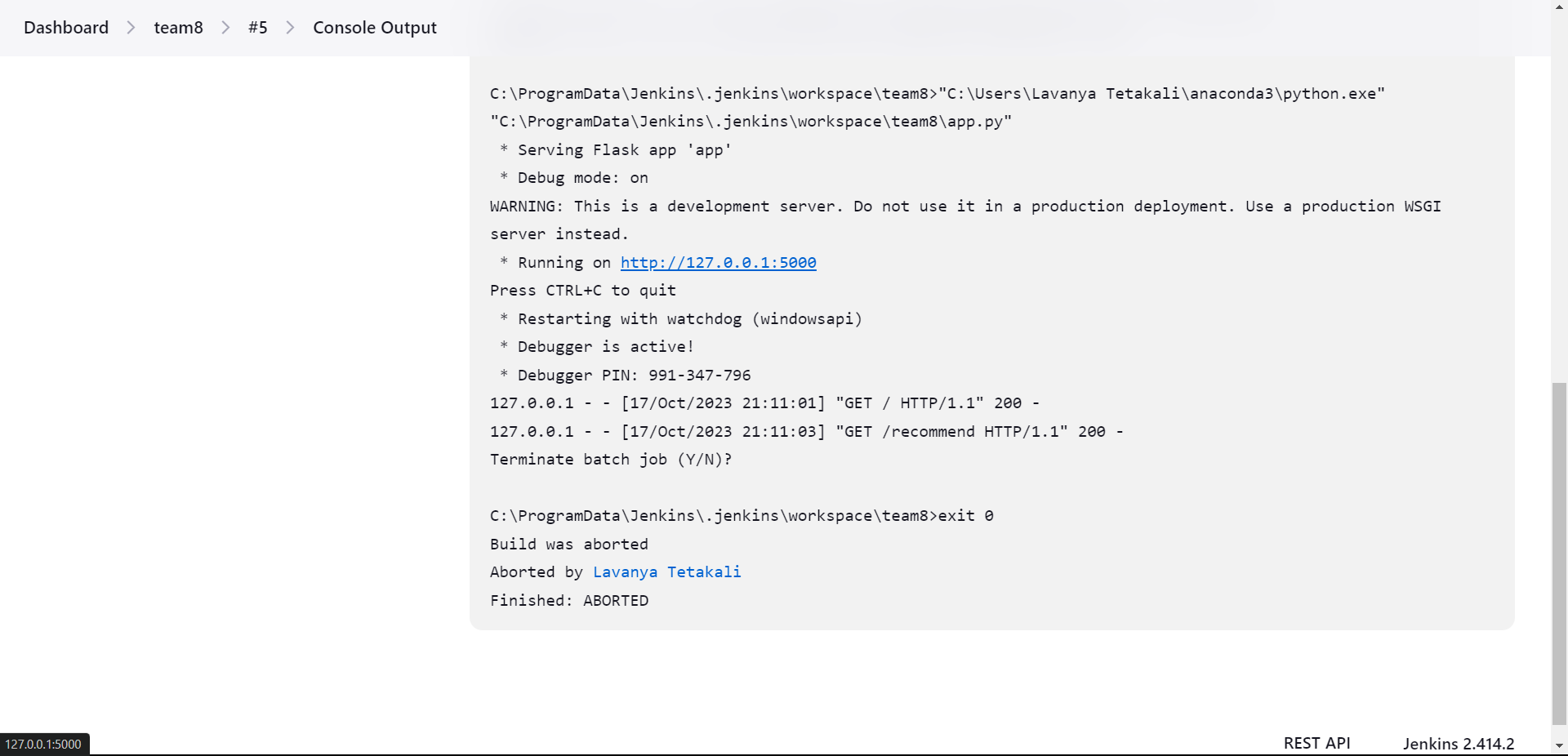


Fig 2.2.5.5 view console output

**2.2.5 Implementation of Jenkins Pipelines:**

**2.2.5.1 What is a pipeline in Jenkins ?**

Jenkins is a popular open-source automation server that allows you to automate various aspects of the software development and deployment process. Jenkins pipelines are a way to define and automate your continuous integration and continuous delivery (CI/CD) workflows within Jenkins.

There are two main types of pipelines in Jenkins:

* **Scripted Pipeline:**

This is the traditional way of defining Jenkins pipelines using Groovy-based DSL (Domain Specific Language). **Declarative Pipeline:**

Declarative pipelines are a more structured and human-readable way to define your CI/CD workflows. They are defined using a simpler syntax and are recommended for those who want to get started quickly and maintain pipelines more easily.

**2.2.5.2 Deploymemt of a job using Declarative Pipeline:**

1. Access the Jenkins Dashboard:

Open a web browser and access the Jenkins web interface by navigating to `http://localhost:8080` or the URL where Jenkins is hosted.

2. Create a New Pipeline Job:

- In the Jenkins dashboard, click on "New Item" or "New Job" on the left-hand side.

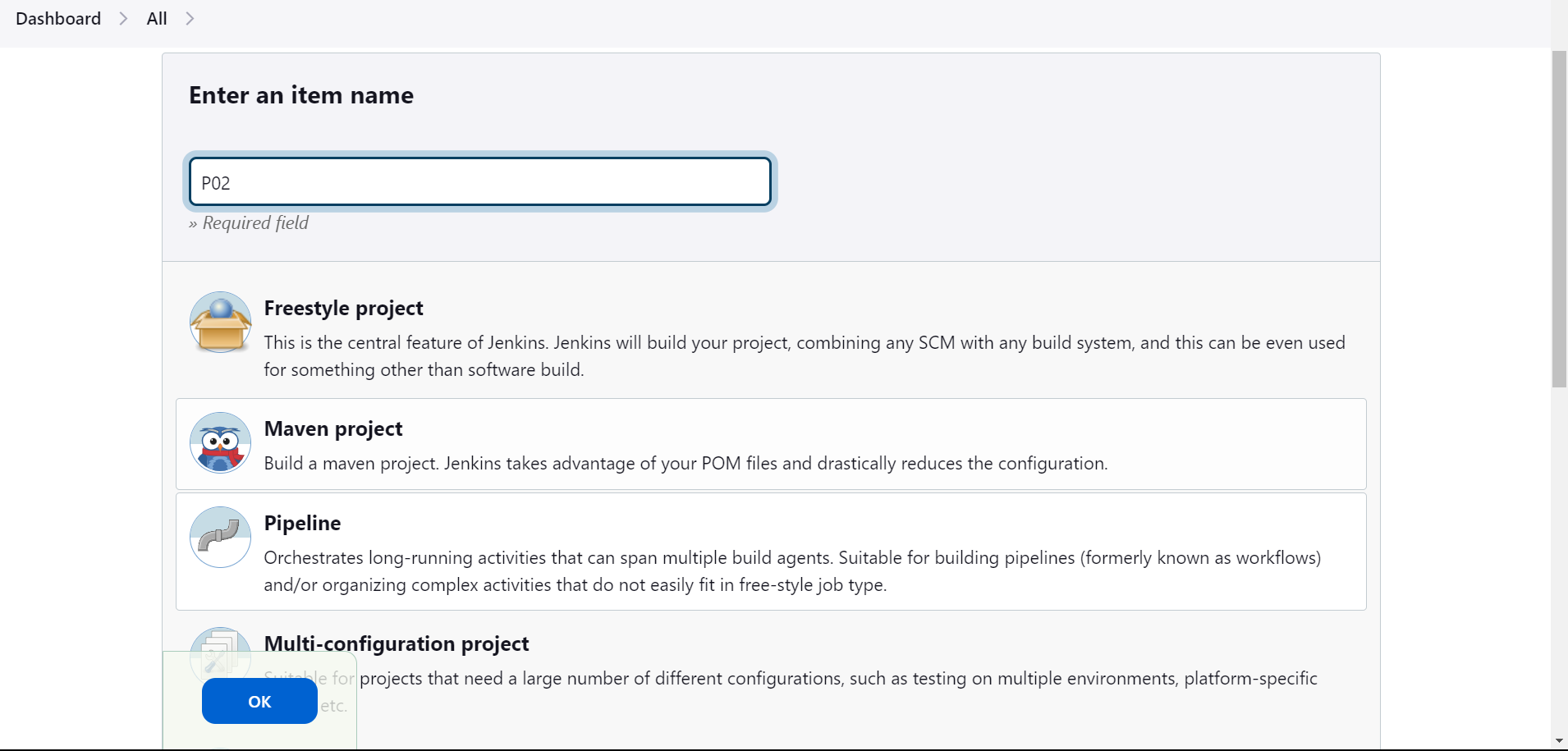


Fig 2.2.6.1 creating pipeline

3. Configure the Pipeline Job:

- Enter a name for your pipeline job (e.g., "MyPipeline").

- Select "Pipeline" as the job type.

- Click "OK" to create the job.

4. Define the Pipeline Script:

- Scroll down to the "Pipeline" section of the job configuration.

- Choose the pipeline script option.

- Paste pipeline script directly in this section:

pipeline {

agent any

stages {

stage('Checkout') {

steps {

// Checkout your source code from version control (e.g., Git)

checkout scmGit(branches: [[name: '\*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/Lavanya-2402/ReadWise.git']])

}

}

stage('build - publish HTML') {

steps {

script {

publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: false, reportDir: 'templates', reportFiles: 'index.html', reportName: 'HTML Report', reportTitles: '', useWrapperFileDirectly: true])

}

}

}

stage('Run app1.py') {

steps {

script {

def pythonExecutable = "C:\\Users\\Lavanya Tetakali\\anaconda3\\python.exe"

def scriptPath = "C:\\ProgramData\\Jenkins\\.jenkins\\workspace\\P2\\app.py"

bat(script: "\"$pythonExecutable\" \"$scriptPath\"", returnStatus: true)

echo("http://127.0.0.1:5000/")

}

}

}

}

post {

success {

echo 'Pipeline succeeded!'

// Add actions to take on success here

}

failure {

echo 'Pipeline failed!'

// Add actions to take on failure here

}

}

}

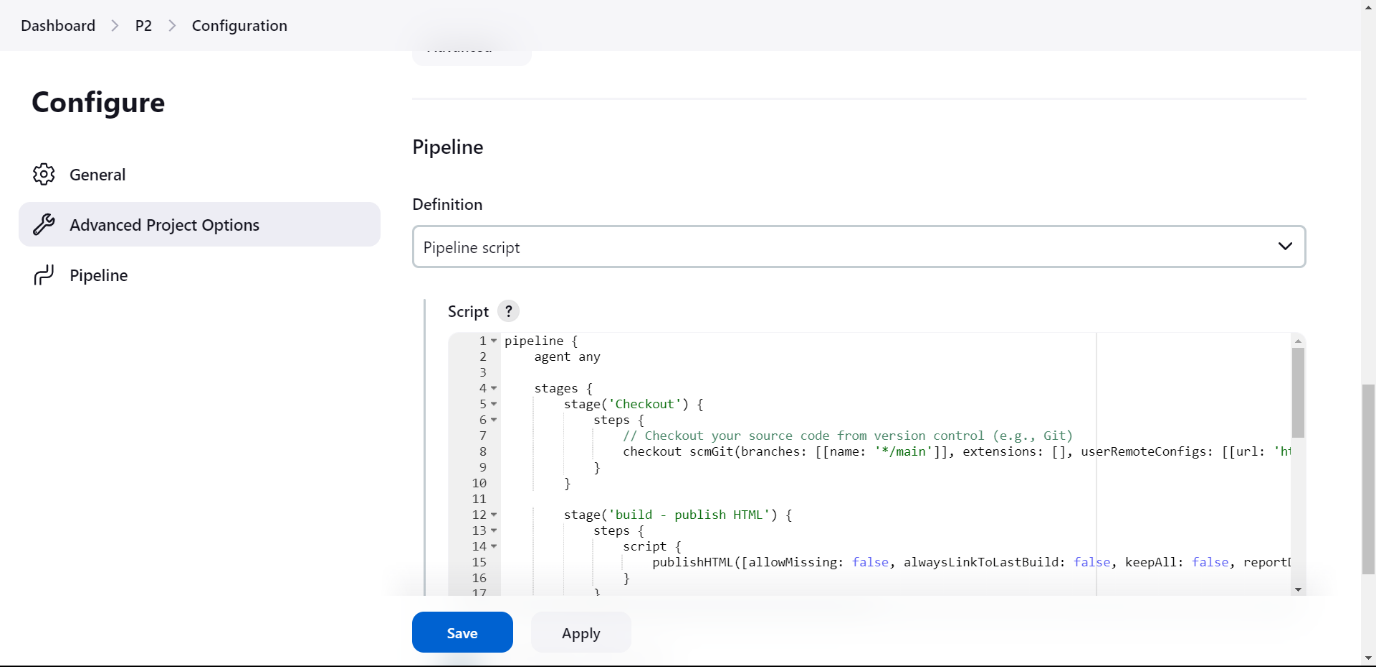


Fig 2.2.6.2 write pipeline script

5. Save the Pipeline Job:

- Click "Save" to save your pipeline job configuration.

6. Manually Trigger the Pipeline:

- After saving the pipeline job, you can manually trigger it by clicking "Build Now."

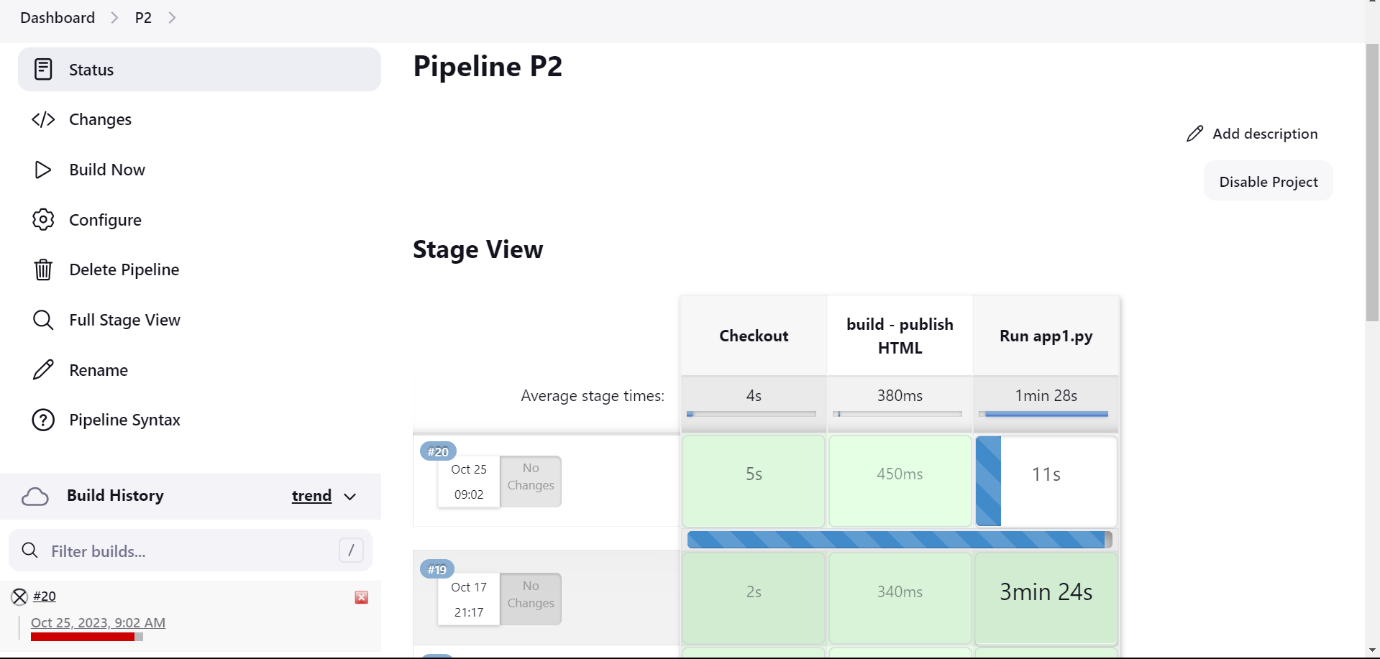


Fig 2.2.6.3 Build Pipeline

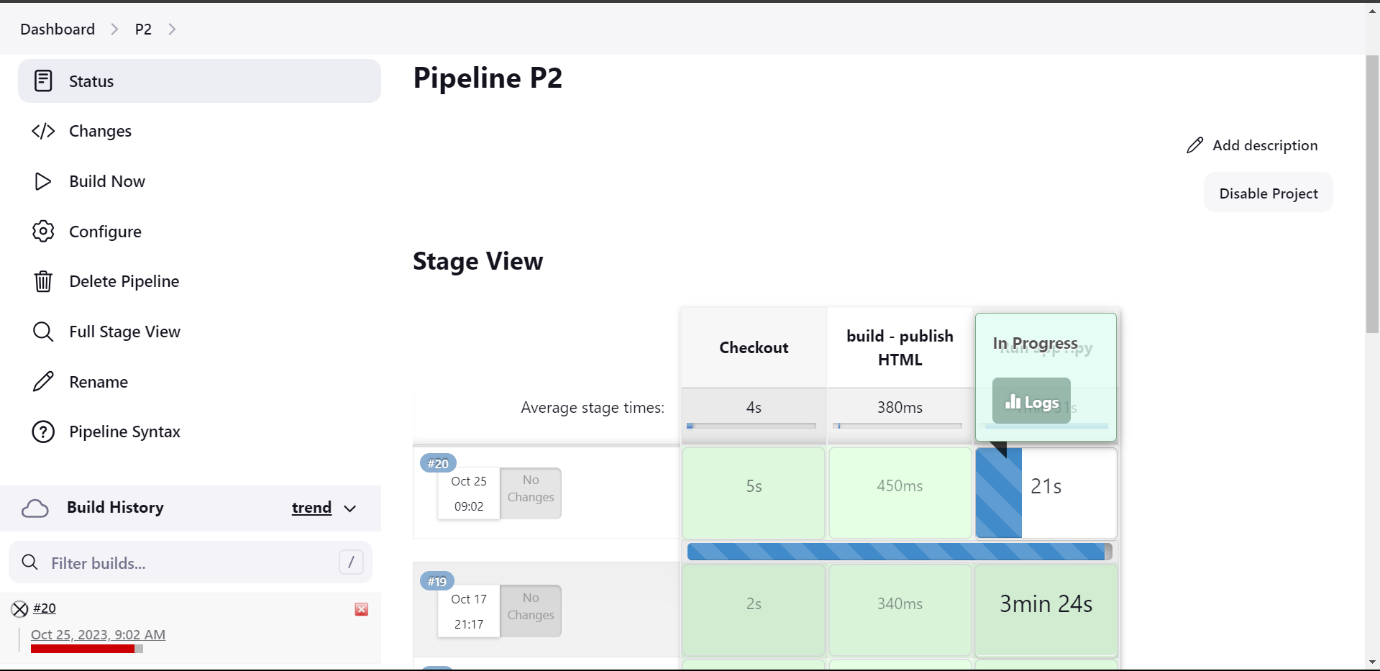


Fig 2.2.6.4 click on logs

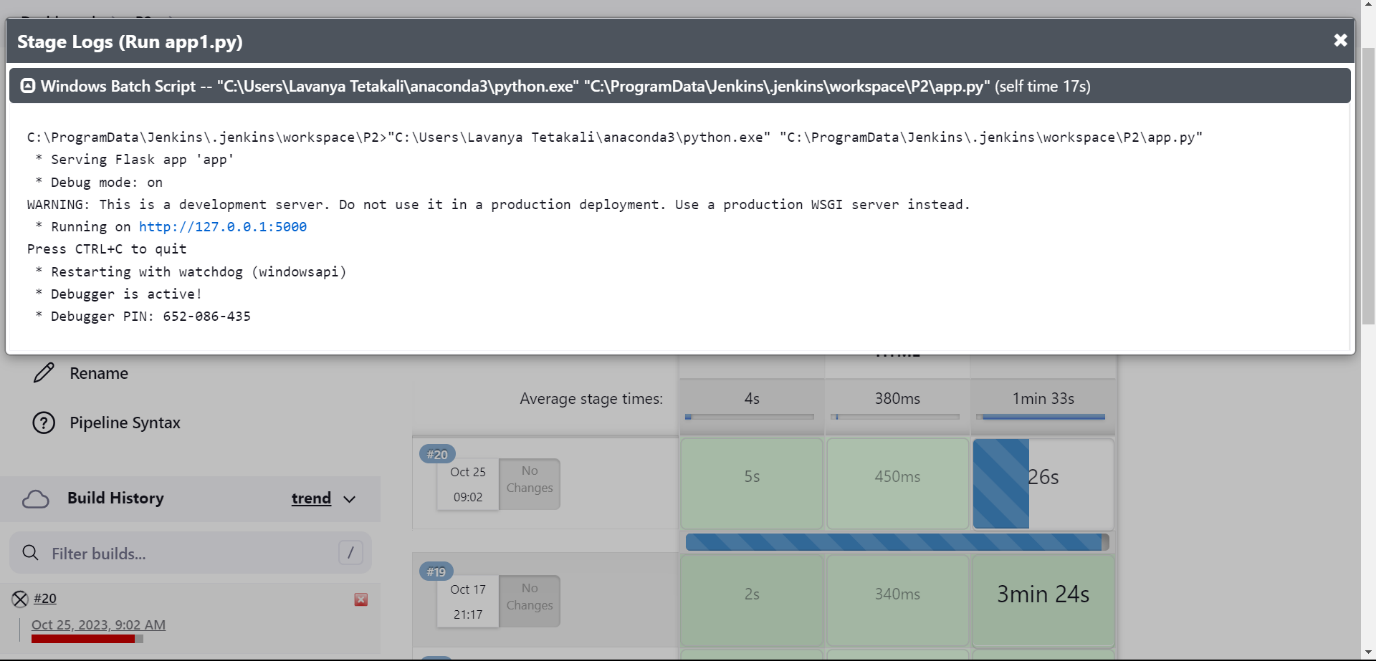


Fig 2.2.6.5 Application URL generated

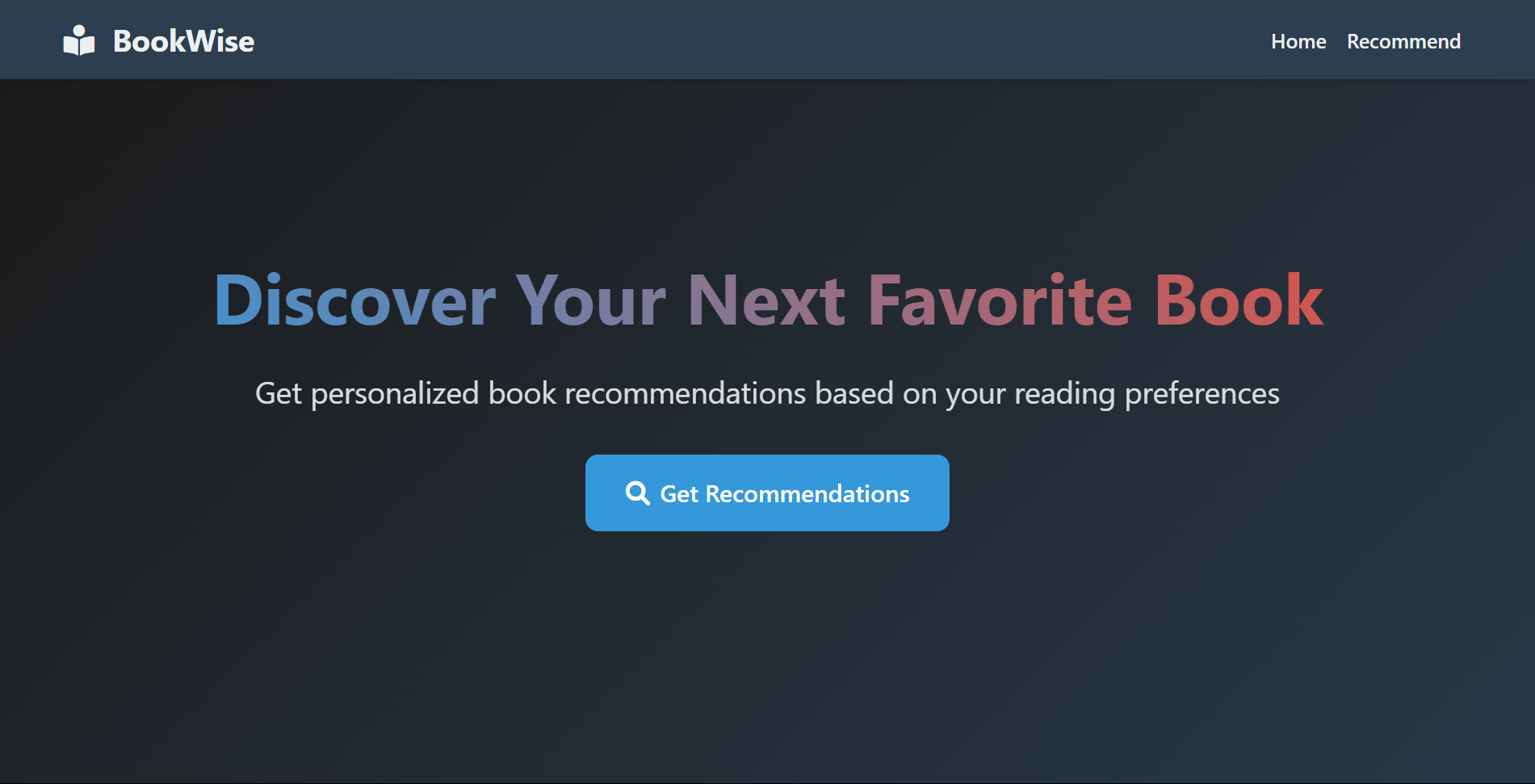


Fig 2.2.6.6 Application Homepage

NOTE: Since Flask is used to create dynamic websites, it's important to note that, by default, the Flask application will continue running indefinitely until manually aborted. In the context of a Jenkins pipeline, you will need to manually stop or abort the pipeline task if you want to terminate the Flask application, as it won't automatically stop on its own.

**3.CONCLUSION**

In conclusion, the book recommendation system represents a significant advancement in the realm of digital content discovery. It leverages an array of cutting-edge technologies, including Flask, Numpy, Pandas, HTML, CSS, and Bootstrap, to provide a user-friendly, data-driven, and personalized book recommendation experience.

1. **Automation:** Jenkins provides powerful automation capabilities for building, testing, and deploying software, while HTML-based event management systems allow for easy customization and automation of event-triggered actions. This reduces manual intervention and minimizes human errors.
2. **Collaboration:** CI/CD using Jenkins fosters collaboration among development, testing, and operations teams by providing a centralized platform for managing code, sharing artifacts, and coordinating deployments. An HTML event management system can enhance this collaboration by offering real-time event tracking and communication.
3. **Speed and Efficiency:** With CI/CD, software can be built, tested, and deployed more rapidly, leading to faster releases and quicker response to customer feedback or market demands. Event management systems help in monitoring the progress of these processes and responding to issues promptly.
4. **Quality Assurance:** Continuous testing and automated quality checks in Jenkins help maintain the overall quality of the software. Event-driven HTML systems can capture and track issues, allowing for prompt resolution and quality improvement.
5. **Scalability:** Jenkins and HTML-based event management systems are highly scalable, accommodating the needs of both small and large development teams and projects. As the software development environment grows, this combination can adapt accordingly.
6. **Transparency:** By integrating an event management system into the CI/CD pipeline, teams gain visibility into the entire software development and delivery process. Stakeholders can monitor progress and track key events in real time, ensuring better transparency and accountability.
7. **Flexibility:** HTML-based event management systems can be customized to suit the specific needs and preferences of your organization, making it easier to adapt the system to unique workflows and event handling requirements.

Combining Jenkins for CI/CD with an HTML-based event management system is a powerful strategy for enhancing software development and deployment processes. This approach promotes automation, collaboration, speed, quality assurance, scalability, transparency, and flexibility. It empowers organizations to keep up with the rapid pace of software development and maintain a competitive edge in the ever-evolving tech landscape.