Draft proposal: “Displaying of Plugin Health Scores”

Google Summer of Code Program 2023 Project Proposal

Harsh Gupta
harshgupta3341@gmail.com
matrix

Table Of Content

Project Abstract

The objective of this project is to present the **Health Scores of Plugins** on plugins.jenkins.io and provide specific details for each plugin. To achieve this, the Plugin Health Score System was created in 2022 to compute individual scores for each plugin, which will be visible on plugins.jenkins.io. Furthermore, we will also implement **filtering options** based on these scores.

Project Description

Jenkins offers over **1800+ plugins** to assist with building, deploying, and automating projects. While many plugins are well-maintained, others are created by newcomers as the community grows. Developers must identify which plugins are in better condition to ensure their project is **built correctly**. The Plugin Health Score System, developed by maintainers in GSoC 2022, tracks the **performance of individual plugins** using data generated by probes and assigns a **score** to each plugin.

The maintainer sets and maintains the foundation of the project. However, one major piece that needs to be added is the implementation of the calculated score, which should be **displayed** on plugins.jenkins.io. By doing so, developers can easily choose the best plugins to use in their projects. This is to be implemented during the period of **GSoC 2023**.

I have chosen to work on this project because I am familiar with its tech stack. As a newcomer to Jenkins, I was initially confused by the large number of plugins available. This confusion led me to the Jenkins community, where I was introduced to the ongoing project idea for GSoC 2023.

Project Deliverables

There are three major areas to look into -

- ➔ **Plugin Health Score System** - link
  - ◆ Project calculating the Health of Plugins using Probes and generating a score for each plugin.

- ➔ **Plugin Site API (This might be Shut Down)** - link
  - ◆ Project handling the backend API of plugins.jenkins.io

- ➔ **Directly fetching the plugin health score in the Plugin-Site (This approach needs to be implemented)**

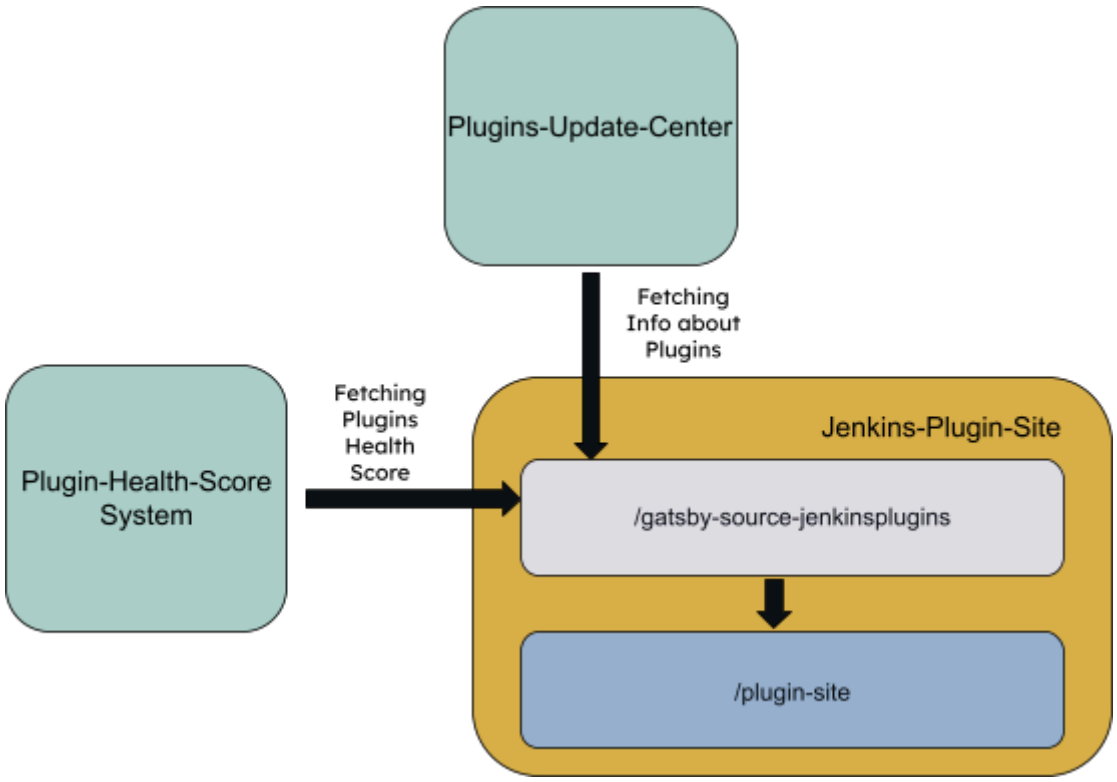
- ➔ **Plugins Site** - link
  - ◆ Frontend project which displays all info about the plugins.

As the Health of Plugins is calculated in another project, significant work is needed on the Plugins site.

1) As mentioned by one of my potential mentors, the Plugins Health Score System project contains an API that provides JSON data for each plugin.

2) This JSON data must be fetched in the plugin site for the implementation of Plugins-Health-Score which can be done using GraphQL queries.

3) Jenkins plugins site using Gatsby to create its frontend functionality which uses Reactjs and GraphQL.



/gatsby-source-jenkinsplugins - Directory present in Plugin-Site repository where data is been fetched.
/plugin-site - Directory present in Plugin-Site repository where frontend of plugins.jenkins.io is written.

Implementation

1. Collectings Plugins Health Score Data

Thanks to my mentors Adrien Lecharpentier and Dheeraj Singh Jodha for building the foundation of the Plugins Health Score project.

Plugins-Health-Score System generates a score for each plugin from the data fetched using Probes(Link), then each score is stored in PostgreSQL.

As Rest API has been implemented by Adrien Lecharpentier in #169 which provides **Plugins Health Score** with some useful details for plugins.jenkins.io integration.

API - plugin-health.jenkins.io/api

Routes	Available
/scores	Yes


```
{
  "aws-java-sdk-sns": {
    "value": 96,
    "version": "1.12.406-370.v8f993c987059",
    "details": [
      {
        "key": "security",
        "value": 1,
        "coefficient": 1
      },
      {
        "key": "repository-configuration",
        "value": 0.7,
        "coefficient": 0.5
      },
      {
        "key": "adoption",
        "value": 1,
        "coefficient": 0.8
      },
      {
        "key": "deprecation",
        "value": 1,
        "coefficient": 0.8
      },
      {
        "key": "update-center-plugin-publication",
        "value": 1,
        "coefficient": 1
      }
    ],
    "timestamp": "2023-03-02T11:32:01.201909Z"
  },
```


Response received from (Link)

Routes	Available
/probes	No

The expected result after implementation

Currently, the API does not contain this route, but it needs to be implemented to use this data to display while building a separate tab for detailed information about each plugin's health score. **This result will contain information about the probes run on the plugin to get the status of the plugin.**

2. API for the frontend application

Here I thought of two implementations

- Modifying the existing Plugin-Site-API
- Directly fetching the plugin health score in the Plugin-Site frontend repository

2.1. Modifying existing Plugin-Site-API

Plugin-Site-API is built to manage the backend of *plugins.jenkins.io* which fetches the info about the plugins.

Plugins information sources include

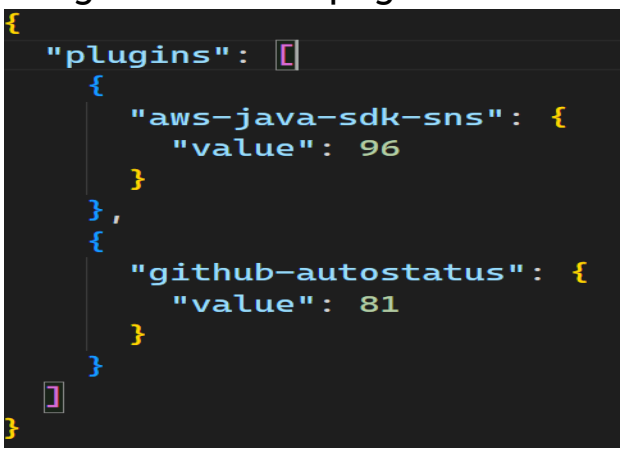
- Jenkins Update Center
    - The main source of plugin data
- Jenkins Stats
    - Installation statistics
- Jenkins Wiki
    - Scrape wiki content

A new method is to be created where we can parse all the data from Plugins-Health-Score and implement it with Elasticsearch.

Possible Routes

1. GET /plugins/healthscore

   Give a score of each plugin that can be used during sorting in the search page



2. GET /plugin/:name/healthscore

   Give details about the requested plugin with its health score, this info can be used in the Health Score section on the plugin page

```
{
  "aws-java-sdk-sns": {
    "value": 96,
    "version": "1.12.406-370.v8f993c987059",
    "details": [
      { "key": "security", "value": 1, "coefficient": 1 },
      { "key": "repository-configuration", "value": 0.7, "coefficient": 0.5 },
      { "key": "adoption", "value": 1, "coefficient": 0.8 },
      { "key": "deprecation", "value": 1, "coefficient": 0.8 },
      {
        "key": "update-center-plugin-publication",
        "value": 1,
        "coefficient": 1
      }
    ],
    "timestamp": "2023-03-02T11:32:01.201909Z"
  }
}
```


Note: There might be a possibility that Plugin-Site-API can be shut down as mentioned by “halkeye” (Gavin Mogan) (Link), thus we can focus more on directly fetching the plugin health score in the Plugin-Site frontend repository.

2.2. Directly fetching the plugin health score in the Plugin-Site frontend repository

The plugin site repository contains a file named utils.js where all the fetching functions are written. is built using Gatsby (a site generator built on top of Node.js using React and GraphQL).

I wrote an **example** function to fetch the data from the Plugin-health-score API, referencing the other functions written there.

```
const fetchPluginHealthScore = async ({ createNode, reporter }) => {
    const sectionActivity = reporter.activityTimer("fetch plugin health score");
    sectionActivity.start();
    const url = "https://plugin-health.jenkins.io/api/scores";
    const json = await requestGET({ url, reporter });
    for (const pluginName of Object.keys(json)) {
        const data = json[pluginName];
        createNode({
            ...data,
            id: pluginName,
            parent: null,
            children: [],
            internal: {
                type: "JenkinsPluginHealth",
                contentDigest: crypto
                    .createHash("md5")
                    .update(`pluginHealth_${pluginName}`)
                    .digest("hex"),
            },
        });
    }
    sectionActivity.end();
};
```

It iterates over the JSON objects containing the plugin health scores and creates a node for each plugin this function is then called in the gatsby-node.js file to fetch and create nodes for plugin health scores.

As GraphQL uses queries to get specific data,
here is the query I wrote to get all the plugin's health scores

```
query MyQuery {
  allJenkinsPluginHealth {
    edges {
      node {
        id
        value
        version
        timestamp
      }
    }
  }
}
```

The resultant gives:

```
{
  "data": {
    "allJenkinsPluginHealth": {
      "edges": [
        {
          "node": {
            "id": "aws-java-sdk-sns",
            "value": 96,
            "version": "1.12.406-370.v8f993c987059",
            "timestamp": "2023-03-06T20:18:11.238627Z"
          }
        },
        {
          "node": {
            "id": "github-autostatus",
            "value": 81,
            "version": "3.6.2",
            "timestamp": "2023-03-02T11:32:05.798917Z"
          }
        },
        {
          "node": {
            "id": "quayio-trigger",
            "value": 88,
            "version": "0.1",
            "timestamp": "2023-03-02T11:32:17.230708Z"
          }
        },
```


as output, there are a total of 1915 nodes.

Through queries, we can get a specific plugin:

```
query MyQuery {
  allJenkinsPluginHealth(filter: {id: {eq: "git"}}) {
    edges {
      node {
        id
        value
        version
        timestamp
      }
    }
  }
}
```


```
{
  "data": {
    "allJenkinsPluginHealth": {
      "edges": [
        {
          "node": {
            "id": "git",
            "value": 96,
            "version": "5.0.0",
            "timestamp": "2023-03-06T21:18:26.173789Z"
          }
        }
      ]
    }
  },
  "extensions": {}
}
```


This query can be used while building tabs for detailed info on each plugin's health score.

Note: There are a lot of changes going on in the plugin site repository, so it is possible that the approach I mentioned while writing this proposal might change during the contribution period.

3. Building the frontend UI

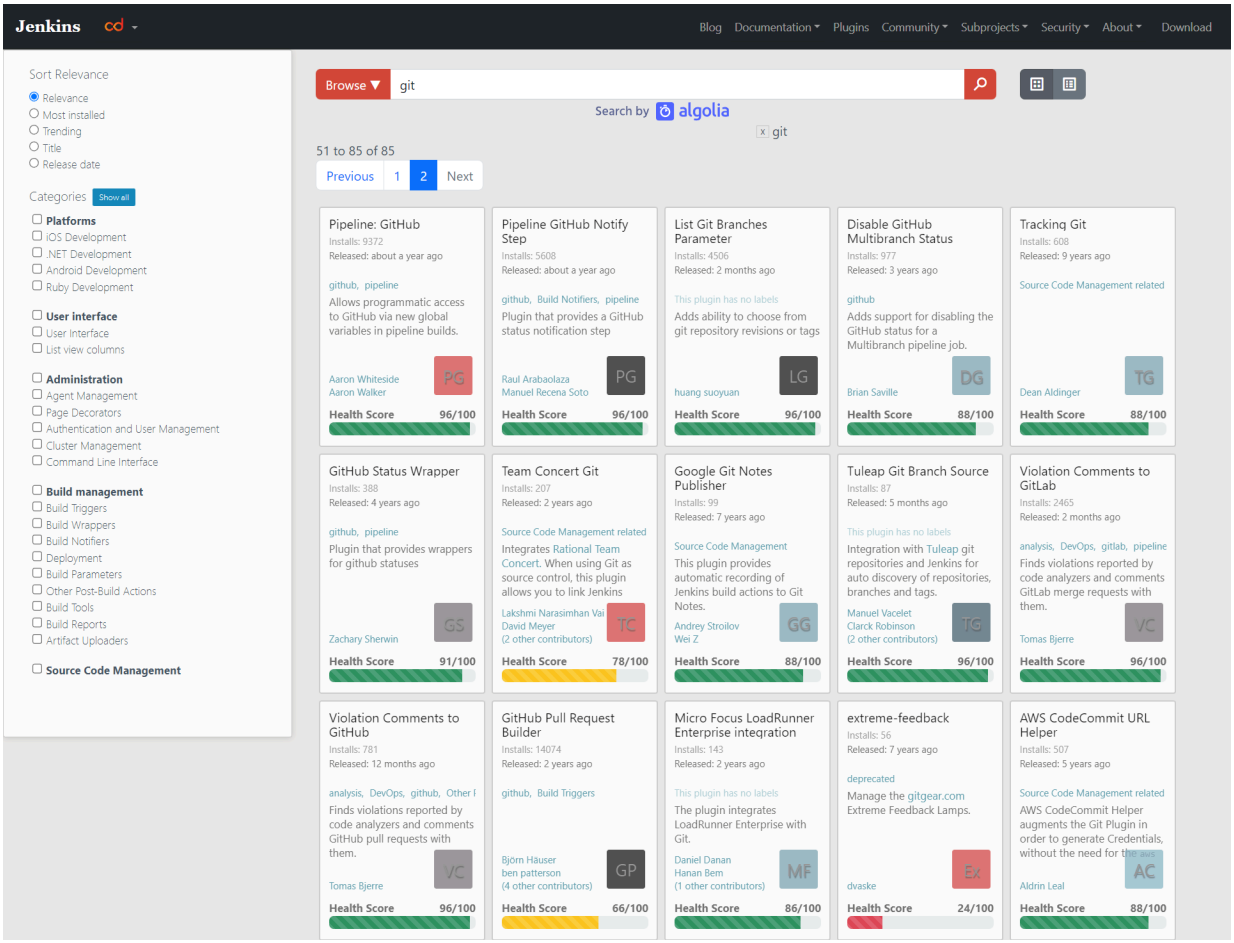
There are 3 areas where the frontend component has to be built:

1. Search Page.
2. Filter Option.
3. Separate Tabs for detailed information about each plugin's health score.

1. Search Page

We need to add a separate component for health score in the plugin.jsx and integrate the fetched data using GraphQL with the respected plugin.

Here is what I implemented using the fetched data I mentioned in **2.2**



2. Filter Option

- This is to be added so that users can sort and choose the plugin with a better health condition for their projects.
- Also, this will be helpful for the developer of the plugin to know the status of his plugin.

3. Separate Tabs for detailed information about each plugin's health score.

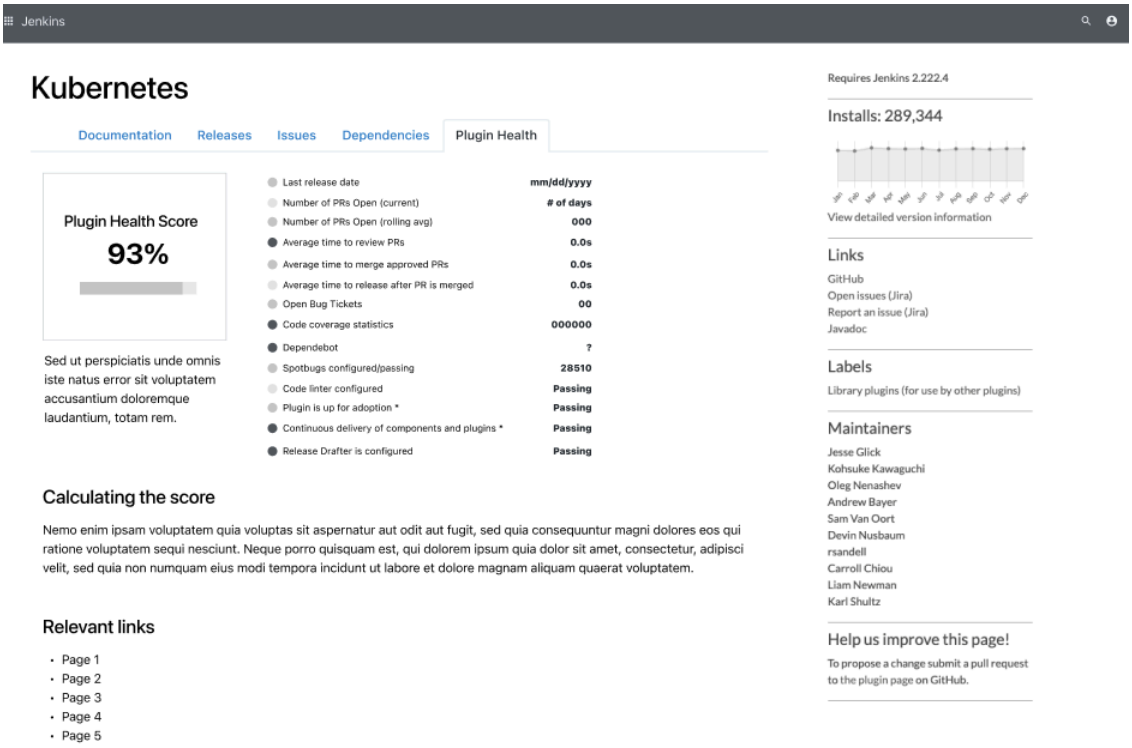
A separate component needs to be added that shows the details of how the health score of a plugin is calculated.

This can be achieved by adding a component in *plugin-site/src/components/* and calling it in *plugin-site/src/components/PluginPageLayout.jsx* where we can give the id as **healthScore** and label it as **Health Score.**

```
const tabs = [
    {id: 'documentation', to: `/${plugin.name}/`, label: 'Documentation'},
    {id: 'releases', to: `/${plugin.name}/releases/`, label: 'Releases'},
    {id: 'issues', to: `/${plugin.name}/issues/`, label: 'Issues'},
    {id: 'dependencies', to: `/${plugin.name}/dependencies/`, label: 'Dependencies'},
    {id: 'healthScore', to: `/${plugin.name}/health/`, label: 'Health Score'}
];
```

Id and label can vary in future development.

Whenever the selected tab id is healthScore we can display the component with detailed information.



Pic credit: Jake Leon

Note: There are a lot of changes going on in the plugin site repository, so it is possible that the approach I mentioned while writing this proposal might change during the contribution period.

Proposed Schedule

March 20 - April 4, 2022 (Application Period)

➔ I will spend this period improving my proposal while putting it for review.

➔ Also will try to interact with the maintainers of plugins.jenkins.io.
- ◆ Many discussions are going on for plugins-site-API, so I will

try to understand more about how they are thinking about the future implementation of API.

April 5 - May 3 (Acceptance Waiting Period)

➔ During this period will continue to understand more about all the areas required during the contribution period.

➔ I will focus on learning more about Maven.

May 4 - May 28 (Community Bonding Period)

➔ 1st or 2nd week of this period might spend on my end-semester exam, but it will not create any significant problems as it is a non-contribution period so I will be active in Gitter and prepare an overview about how I can complete each week's task during the coding period

May 29 - July 9 (Coding Period 1)

The main goal for this phase will be to complete the API building part after discussing with plugin site maintainers how they want this feature to be built.

➔ Complete the writing of the functions and test them so that they can not degrade the performance of the site.

➔ During the last week of this phase, I will spend on testing the build and preparing my presentation for Midterm Evaluation.

July 10 - July 13 (Midterm Evaluation)

➔ Note the feedback from the mentors and org admins and start building an overview of the next phase.

July 14 - August 21 (Coding Period 2)

The goal of this phase will be to build the frontend components of plugins.jenkins.io.

➔ Fetch the data from the API built and integrate it with the components.

➔ Implement the filtering based on the fetched data on the search page.

➔ Make the site more responsive and user-friendly.

➔ At the end of this phase, I will start testing the implementations.

August 21 - August 28 (Final Coding Period for Standard Route)

➔ Continue with the testing.

➔ Given that project ideas **1.“Add probes to Plugin Health Score”** and **2.“Displaying of Plugin Health Score”** fall under the same category, I will be collaborating with the contributor assigned to project 1 to work on the documentation of Plugin Health Score.
This collaboration aims to provide a comprehensive understanding of how the Plugin Health Score is calculated and the various aspect in which it is displayed on *plugins.jenkins.io*. This documentation will accurately reflect the intricacies of Plugin Health Score and its impact on the Jenkins ecosystem.

➔ Start preparing my presentation.

Future Improvements

➔ Future improvement can be displaying Plugin Health Score in Jenkins Plugin Manager.

➔ Work with the plugin site maintainers to remove the usage of plugin-site-API and merge the plugin health score feature built with the ongoing development in the plugin-site repo.

➔ Discuss with mentors and org admins how we can bring transparency to the document and make developers and newcomers understand the background calculation of the plugin health score.

Continued Involvement

➔ I wanted to keep contributing to Jenkins as CI/CD interests me a lot and also the Jenkins community is incredibly supportive towards newcomers, They are continuously encouraging contributors to present their idea and also guiding them to make it possible, As I see many beginners are coming to Jenkins to learn and contribute, especially during Hacktoberfest and the GSoC period. I wanted to guide them the same way I get guidance from Jenkins's mentor.

➔ I will spread awareness among my local communities in conferences and meetings about the opportunities and benefits of involving yourself in open-source communities.

➔ I want to contribute to jenkinsci/jenkins which I will do when I am confident enough with some technologies under Java.

Conflict of Interests or Commitment(s)

I don't have any significant commitments during the GSoC period. As I'm a second-year student, I just have my end-semester exams during 1st or 2nd week of May which will last around 10-12 days. Also, it will be a community bonding period so there will not be much code contribution at that time. I'll be active in the community and prepare an overview for the coming contributions days.

After my exams, I can work approximately 20 hours a week, most of it will be on weekends around 4-6 hours/day, and 2-3 hours/day during weekdays.

Major Challenges Foreseen

// to complete

➔ There are a lot of changes going on in the plugin site repository, so I have to keep myself updated with the changes made by maintainers and discuss the best possible implementation strategy while maintaining the contribution guidelines.

References

- Google
- Jenkins Project Idea
- Jenkins Plugin Site
- Plugin Health Score System GitHub
- GSoC Timeline
- Past GSoC Proposal

Relevant Background Experience

Since High school, I got into programming and started building a railway management system using C++ (at that time I was unaware of git/github so I don't have any link to that project). Since last year I have majorly focused on learning a particular language from where I can start building small projects and learned about Git/Github, from there I have kept track of every personal/open-source project I have worked on.

- **jenkins-infra/plugin-site**
  - Fixed plugin page tab URLs, so they get matched and marked active properly while discussing its further improvement as a lot of changes, are going to be made in the upcoming times(Link).

Bug fix	#1296

- **Open-Source (Hacktoberfest 2022)**

JavaScript Code	#11

JavaScript Code	#14
Documentation	#91
Documentation Correction	#92

- **Full-Stack development**
  - Building a blog publishing website using ReactJs, NodeJs, ExpressJs, and MongoDB - link

- **Frontend development**
  - Built a frontend UI using Atri Framework - link
  - Best built to practice ReactJs built clone, built Disney clone - link
  - Made a note-taking app - link

I just started with the Jenkins ecosystem, and have been going through the codebase where I can provide contributions according to the tech stack I'm familiar with.

Currently, I'm focused on the project idea "Displaying Health Score" where I can take the project to the end. For this project, I have gone through learning Gatsby and am familiar with Java (for intermediate development, which I can learn more about during the period).

Personal Information

Hi, my name is Harsh Gupta, I live in Bilaspur, Chhattisgarh, India. I am a second-year student pursuing a B.Tech in Computer Science and Engineering from Shri Shankaracharya Technical Campus (SSTC) Bhilai. I began my journey with programming during high school, where I was introduced to C++ and developed a railway management system.

My interest in open-source began during my freshman year when **Kunal Kushwaha** introduced me to the concept. I started by learning the necessary technology stack for contributing to open-source projects, such as Git and GitHub. As a beginner, I searched for projects suitable for newcomers, and during **Hacktoberfest 2022**, I contributed to several JavaScript projects and DSA solutions.

While exploring DevOps, I discovered Jenkins plugins used for CI/CD and learned that it was an open-source project. As a participant in GSoC 2023, I plan to immerse myself in the Jenkins codebase and make significant contributions with the guidance of experienced professionals. The Jenkins community is incredibly supportive of newcomers. I aspire to give back to the community once I am confident in my knowledge of the Jenkins ecosystem.

I am always willing to assist others, whether they are my peers or juniors, with their queries and doubts regarding the tech and open-source ecosystems. I find that the Jenkins community has much to offer in this regard, with many experienced professionals willing to help and grow the community.

Open Source Contribution

- **jenkins-infra/plugin-site**
    - Fixed plugin page tab URLs, so they get matched and marked active properly while discussing its further improvement as a lot of changes, are going to be made in the upcoming times(Link).

Bug fix	#1296

- **Hacktoberfest 2022**
    - Started with some basic JavaScript projects as I was new to knowing about open-source.

JavaScript Code	#11
JavaScript Code	#14
Documentation	#91
Documentation Correction	#92

- **Codinasion / program (DSA-based Repo)**
    - I was practicing DSA at that time using Java, so I decided to add some solutions to the code.

Java Code	Merged

Language and Other Skill sets

- Programming Language
    - C/C++ (Intermediate)
    - Java (Intermediate)
    - Go (Beginner)
- Scripting Language
    - JavaScript (Intermediate)
    - Typescript (Beginner)
- Frameworks
    - ReactJs (Intermediate)

    - NodeJs (Intermediate)
    - Bootstrap (Intermediate)
    - TailwindCSS (Intermediate)
- Tools
    - Git/GitHub (Intermediate)
    - Docker (Beginner)
    - Linux (Intermediate)
- Database
    - MongoDB (Intermediate)
    - MySQL (Intermediate)
    - PostgreSQL (Beginner)

Reference Links and Web URLs

- **LinkedIn -** Harsh_Gupta
- **Twitter -** I_harsh_I
- **Matrix -** @harsh3341
- **Timezone -** Indian Standard Time (IST), UTC+5:30
- **Location -** Bilaspur, Chhattisgarh, India