

# Automated Download Speed Testing For Download.Kiwix.Org

GSoC 2024 Proposal @Kiwix

## Personal Details

### Identity Information

- Name: Meer Ismail Ali
- University: National Institute of Technology Karnataka, Surathkal (NITK)
- Email: meerismailali.211ec125@nitk.edu.in, meeraliakhil@gmail.com
- Github: <https://github.com/MeerIsmailAli>
- LinkedIn: <https://www.linkedin.com/in/meer-ismail-ali-623334220/>
- TimeZone: IST (UTC + 5:30)
- Location: Mangaluru, India

### Patch Requirement

S no	Description	Link
1.	Hovering UI change : styling for background of logo column and download/open column while mouse hovering over the row.	<a href="https://github.com/kiwix/kiwix-desktop/pull/1062">https://github.com/kiwix/kiwix-desktop/pull/1062</a>

## Background

### Education

I am currently a pre-final undergraduate student at the National Institute of Technology Karnataka, Surathkal (NITK). I am pursuing a degree in Electronics and Communications Engineering and have undertaken courses like Computer Programming, Computer Networks.

I began my undergraduate studies in 2021 and have mainly worked with C and C++ languages

and undertaken projects with Python, Javascript and some other languages. I am primarily interested in Computer Networking and Web development.

## Work

S no.	Name	Domain	Description
1.	Pedes Website	Web App: Django, tailwind CSS	Uses MYSQL and Django as framework and database , with tailwind for the frontend
2.	Stock Analyzer	React JS, rechart library	Fetches real time candle data and plots them with visual libraries to enable stock viewing across a span

## Experience with Org

I am new to kiwix ecosystem and working on its source code. I have learnt a lot about kiwix code structure and especially the kiwix-desktop app as I made my initial contributions and prepared this proposal.

## Open Source Experience

- Started on working for a website for an IEEE conference taking place in my university where I had to collaborate with multiple members and work with the code on github. Here is the [link](#).
- I made contributions to the kiwix repository while preparing my proposal where I compiled the kiwix-desktop app on my local machine and started to understand the codebase since I was unfamiliar with Qt applications.

## Why Me

I am all excited about open-source projects and contributions. I started with open-source contributions last year and developed an interest in it. I am very passionate about exploring different parts of codes, making changes, looking for bugs and contributing, if suitable. During my sophomore year, I started learning about computer networking basics and was fascinated with it.

A few days back, when I came across the problem statement to build a solution for testing the websites download speed across multiple locations around the globe I was very intrigued and started researching solutions for the same.

I am very hard-working and dedicated to my work and always like to learn new things. With GSoC, I get the opportunity to work under leading open-source contributors and learn a lot from their knowledge and experience.

# Project Details

## Introduction

This plan outlines the approach for evaluating download speeds and Kiwix download simulations from various locations using a single cloud machine with WireGuard. The project aims to gather data on performance metrics without deploying testing machines in each location.

## Approach

### Aim

The primary aim of this project is to establish a system for:

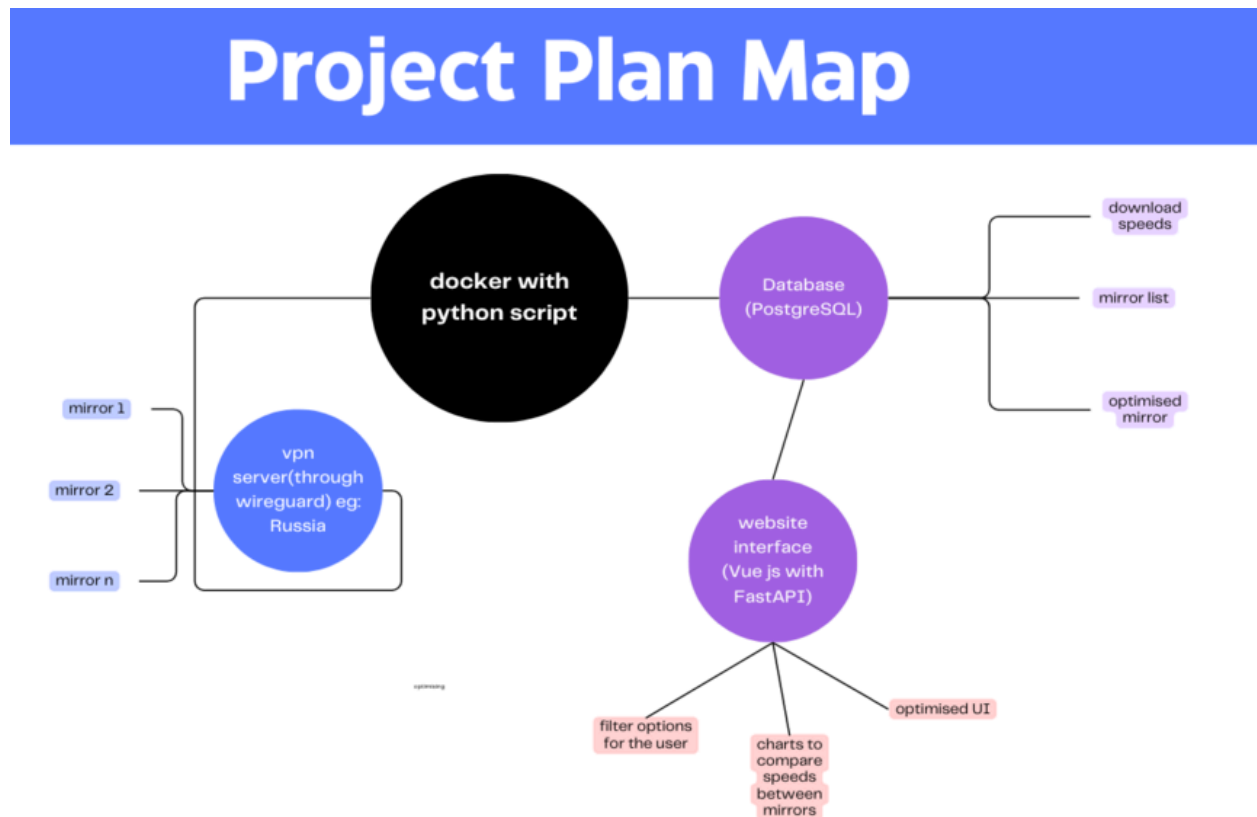
- \* Conducting download speed tests.
- \* Simulating Kiwix downloads.
- \* Collecting and storing performance data from various geographical locations.

## Proposed Approach

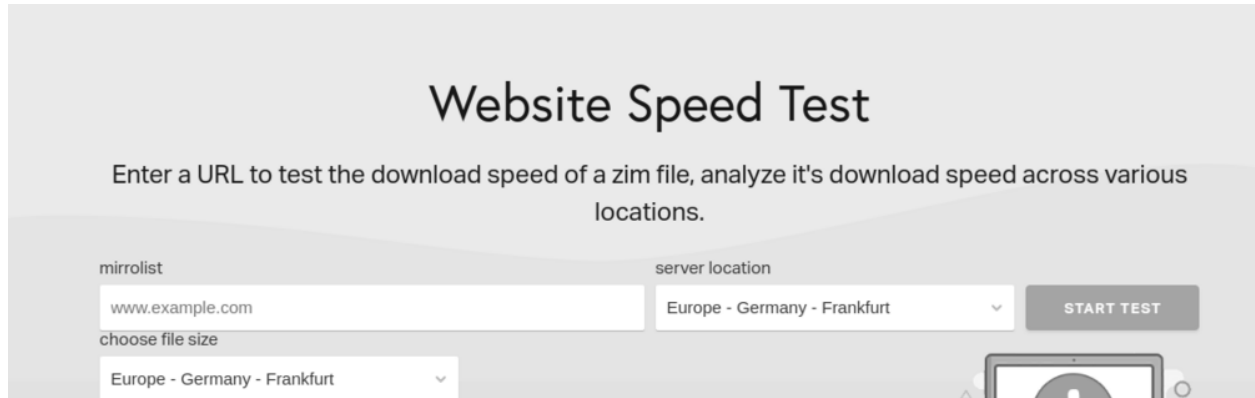
- Our aim is to deploy a script to run on a particular machine(cloud) this script must be able to connect to a vpn provider(nordVPN) through wireguard protocol .
- The VPN provider must provide servers across the globe.
- For each targeted region of interest, we initiate connections to the corresponding or proximate server through the VPN and then run the script to download a zim file iterating through all the mirror lists.
- The script will then output the avg speed tests of different mirrors in that region. which is saved in the database (PostgreSQL).
- We periodically run this script at certain intervals and update the database accordingly. This will then be updated on the database.

- Then to access this data we want to create API endpoints with FastAPI and the front end with vue js to create a web app to make the interface. The project will be running inside a container ensuring consistency, portability, and scalability.

## Project Map



## UI Initial Prototype



The image shows a web interface titled "Website Speed Test". Below the title, there is a text prompt: "Enter a URL to test the download speed of a zim file, analyze it's download speed across various locations." The interface includes a text input field labeled "mirrolist" containing "www.example.com", a dropdown menu labeled "server location" with "Europe - Germany - Frankfurt" selected, and a "START TEST" button. Below these, there is another dropdown menu labeled "choose file size" also with "Europe - Germany - Frankfurt" selected. A small icon of a laptop is visible in the bottom right corner.

## Plan

### Deliverables

- A python script that tests download speed from various locations on earth.
- A database to store the test results.
- A web-based interface to view the results.
- Documentation for how to set up and use the solution.

### ***Milestone 1: Core Functionality with Docker, PostgreSQL, and NordVPN WireGuard***

This milestone focuses on establishing the foundation for the project by setting up core functionalities within a single Docker container.

#### Tasks:

##### 1. Dockerfile:

- Base image: Python (e.g., `python:3.9`)
- Installing dependencies:
  - Libraries for script execution (e.g. `requests`)
  - PostgreSQL client (`psycopg2`)
  - NordVPN command-line tool (`nordvpn`) - Following NordVPN's guide to install on Linux;

- Copy script files: Place your test automation scripts (`test_script.py`) and any dependencies within the container.
- **NordVPN Configuration:**
  - Include the NordVPN credentials and WireGuard configuration files within the container. These files might need to be stored with restricted access permissions inside the container.
- 2. **PostgreSQL Database Setup:**
  - Utilize a pre-built PostgreSQL Docker image alongside the script runner container.
  - Configuring the database during container creation using environment variables for username, password, and database name.
- 3. **Database Integration:**
  - Within the script runner container, configure the PostgreSQL client to connect to the database using:
    - Hostname/IP address of the PostgreSQL container within the Docker network.
    - Database credentials stored as environment variables.
- 4. **Script Functionality:**
  - Developing the script (`test_script.py`) to perform the following tasks:
    - **Connect to NordVPN using WireGuard:** Utilize the `nordvpn` command-line tool to connect to a desired NordVPN server based on the script logic (e.g., location selection).
    - **Run Download Speed Tests:** Coding a python script to download files and measuring speed to perform download speed tests and store the results.
    - **Kiwix Download Simulation:** Simulate a Kiwix file download process and measure the download time.
    - **Data Collection and Storage:** Connect to the PostgreSQL database and insert the collected test results (location, VPN server details, speed test results, Kiwix download time).
- 5. **Script Scheduling:**
  - Utilize cron jobs on your cloud platform to schedule the script runner container to start periodically (e.g., hourly).

## **Milestone 2: Visualization and Location Testing**

- **Objective:** Enhance the project by adding a web interface and performing initial location testing.
- **Tasks:**
  - **Web Interface:** Develop a basic web interface using VueJS to visualize the collected data in the database. This might include charts or tables showing download speeds and Kiwix download times across locations.
  - **Database Interaction:** Utilize a database library like `psycopg2` within the FastAPI application to connect to the PostgreSQL database. Implement functions within the API endpoints to:

- **Receive data** from the script runner container via the `/results` POST endpoint.
- **Parse and validate** the received data.
- **Execute SQL queries** to insert the validated data into the appropriate database tables.
- **Respond** with a success or error message to the script runner container.
- **VPN Location Testing:**
  - Choose a commercial VPN (NordVPN) provider offering downloadable WireGuard configuration files.
  - Obtain WireGuard configuration files for a few initial locations.
  - Modify the script to handle connecting to different VPN locations using the corresponding configuration files.
  - Conduct initial testing from these locations and verify data collection for each location.

## Milestone 3: Refinement and Optimization with Docker Compose

This milestone is to focus on polishing the project for easier deployment and management. To integrate Docker Compose:

### Tasks:

1. **Documentation:**
  - Create comprehensive documentation covering:
    - Project setup instructions (dependencies, environment variables).
    - Script functionality details (download tests, Kiwix simulation, data collection).
    - Web interface usage guide (visualizing results, potential filtering options).
    - Overall project architecture and technology stack.
  - Docs with tools like Sphinx or MkDocs to generate user-friendly documentation.
2. **Docker Compose:**
  - a `docker-compose.yml` file to define the multi-container application:
    - **Services:**
      - **web:** This service will define the Vue.js web interface container.
      - **script-runner:** This service defines the container running the Python script for tests.
      - **database:** This service defines the PostgreSQL database container.

- **Volumes:** (Optional: since the downloaded .zim files themselves need not be saved, we only need the download performance metrics) Define volumes to persist data like web interface assets or script logs outside the containers.
- **Networks:** Create a Docker network for all containers to communicate with each other.
- **Environment Variables:** Specifying environment variables for sensitive information like database credentials or API base URL (used by both script and web interface).

## Milestone 4: Testing ,Finalization and Handover

This milestone focuses on finalizing the project including testing, ensuring its quality, and preparing it for handover or deployment.

### Tasks:

#### 1. Handover Meeting :

- During the meeting, demonstrate the project functionalities, discuss deployment procedures, and address any handover-related questions with the mentor.
- Discussing with the mentor and reviewing the project details. Adding additional functionalities or modifying existing code/features to be done this period.

#### 2. Comprehensive Testing:

- Conducting thorough testing of the project across various aspects:
  - **Script Functionality:** Verify that download speed tests, Kiwix download simulation, and data collection work as expected across different VPN server locations.
  - **Web Interface:** Test all functionalities of the web interface, including data visualization, filtering options (if applicable), and overall user experience.
  - **Database Integration:** Ensure seamless data transfer between the script runner and the database.
- Using different Kiwix mirror lists in your testing to identify the most reliable options for different regions.
- Document any testing procedures and results for future reference.

#### 3. Project Clean Up:

- Remove unnecessary files, temporary configurations, or debugging code from your project directory.
- Organizing the codebase for clarity and maintainability.

#### 4. Final Docker Image:

- Based on your Docker Compose configuration (from Milestone 3), create a single, optimized Docker image containing all necessary components



- (web app, script runner, dependencies).
- Consider using multi-stage builds to create a smaller final image by removing development dependencies during the final build stage.
- Utilize tools like Docker build cache to optimize image creation time.
- 5. **Delivery Package:**
  - Prepare a comprehensive delivery package containing:
    - **Project Documentation:** Include all relevant documentation created in Milestone 3 (setup instructions, user guides, etc.).
    - **Docker Image:** Provide the final Docker image for easy deployment.
    - **Readme File:** Create a comprehensive `README.md` file outlining project overview, installation instructions, usage notes, and any additional information for future users.

## Testing Methodology

The testing approach will utilize a combination of integration testing and system testing, simulating real-world user scenarios.

### Schedule:

1. Test environment setup: Verify access to Docker, Python, NordVPN account, PostgreSQL database, and testing tools.
2. Script execution testing: Verify script functionality within Docker container, Kiwix file download, and download speed measurement.
3. VPN connectivity testing: Verify successful connection to NordVPN servers using WireGuard across different geographical locations.
4. Kiwix mirror list iteration testing: Ensure script iterates through provided Kiwix mirror lists and attempts downloads from multiple mirrors within a region.
5. Database integration testing: Verify successful connection to PostgreSQL database and data insertion of collected download speed results.
6. FastAPI API Gateway testing: Test API functionality for retrieving download speed data from the database, focusing on basic data fetching requests.
7. Vue.js web interface functionality testing: Verify successful data visualization of download speed results. Test core user interaction features like filtering data (e.g., by region or date).
8. Basic user experience (UX) evaluation: Assess the web interface for clarity, ease of use, and navigation.
9. Test report creation: Document test results with comprehensive details, pass/fail

status, and identified issues/bugs.

10. De-briefing and next steps: Discuss test results, prioritize bug fixes based on severity, and define next steps for further testing or enhancement.

## Testing Data:

- Utilize sample Kiwix mirror lists for testing purposes.
- Generate moderate amounts of mock downloadTest environment setup: Verify access to Docker, Python, NordVPN account, PostgreSQL database, and testing tools.
- Script execution testing: Verify script functionality within Docker container, Kiwix file download, and download speed measurement.
- VPN connectivity testing: Verify successful connection to NordVPN servers using WireGuard across different geographical locations.
- Kiwix mirror list iteration testing: Ensure script iterates through provided Kiwix mirror lists and attempts downloads from multiple mirrors within a region.
- Database integration testing: Verify successful connection to PostgreSQL database and data insertion of collected download speed results.
- FastAPI API Gateway testing: Test API functionality for retrieving download speed data from the database, focusing on basic data fetching requests.
- Vue.js web interface functionality testing: Verify successful data visualization of download speed results. Test core user interaction features like filtering data (e.g., by region or date).
- Basic user experience (UX) evaluation: Assess the web interface for clarity, ease of use, and navigation.
- Test report creation: Document test results with comprehensive details, pass/fail status, and identified issues/bugs.

- De-briefing and next steps: Discuss test results, prioritize bug fixes based on severity, and define next steps for further testing or enhancement.d speed data to simulate realistic test results.

## Conclusion:

This plan facilitates a focused and intensive testing effort to validate the core functionalities of the system and identify critical bugs before wider deployment. Following this plan and addressing identified issues will ensure a more robust and reliable system for further development and user testing.

## Timeline Summary

Timespan	Activity
May 1 - May 26	Community Bonding Period: <ul style="list-style-type: none"> <li>• Discussing the plan with the mentors and community to select the best possible strategy to implement the project.</li> <li>• Discuss the challenges involved in the selected Implementation with the mentor. Particularly discuss the API structure to the postgresSQL database in order to plan the final API design to allow different contributors access to data.</li> <li>• Get familiar with the required components of the project(especially vueJS) by reading documentation and getting feedback from the mentor on the technical aspects.</li> </ul>
May 27 - June 8 (2 weeks)	<ul style="list-style-type: none"> <li>• Set up a Docker container for executing Python scripts.</li> <li>• Begin development of the Python script for downloading files, calculating download speeds, and interacting with NordVPN/WireGuard configurations.</li> <li>• Design and implement secure configuration files for establishing connections to NordVPN servers utilizing WireGuard.</li> </ul>
June 9 - June 21 (2 weeks)	<ul style="list-style-type: none"> <li>• Set up a PostgreSQL database within a Docker container to store collected test results.</li> <li>• Initiate recording test data for different VPN locations and their corresponding mirror list download speeds. This data will be stored in the PostgreSQL database.</li> </ul>

June 22 - June 29 (1 week)	<ul style="list-style-type: none"> <li>● Integrate cron jobs with the script inside docker to automatically trigger the script execution at predefined intervals.(every hour or so)</li> </ul>
June 30 - July 5 (1 week)	<ul style="list-style-type: none"> <li>● Analyze and identify potential performance issues impacting VPN connections (e.g., server load, network congestion).</li> <li>● Implement alternative solutions to overcome identified bottlenecks: <ul style="list-style-type: none"> <li>○ Reconnect to the same server for potential transient issues.</li> <li>○ Utilize different servers within the same region to test download speeds in a broader scope.</li> </ul> </li> </ul>
Phase 1 evaluation	
July 6 - July 12 (1 week)	<ul style="list-style-type: none"> <li>● Evaluate and choose a suitable web framework (VueJS) to construct the user interface.</li> <li>● Integrate visual libraries, such as Vue.js and Chart.js, to create clear and interactive charts depicting download speeds across time.</li> <li>● Implement functionalities allowing users to select files for download (based on data availability in the database) and interact with visualizations.</li> </ul>
July 13 - July 19 (1 week)	<ul style="list-style-type: none"> <li>● Develop an API (e.g., using FastAPI) to act as a communication bridge between the web application and the database. This API will facilitate data exchange.</li> </ul>
July 20 - July 26 (1 week)	<ul style="list-style-type: none"> <li>● Conduct a thorough code review to optimize performance, improve readability, and address any identified issues.</li> <li>● Utilize tools like MkDocs to create comprehensive documentation covering project setup, functionalities, API usage, and overall architecture.</li> </ul>
July 20 - July 26 (1 week)	<ul style="list-style-type: none"> <li>● Share the project code with mentors or peers to obtain valuable feedback on functionality, efficiency, and best practices.</li> <li>● Based on the received feedback, refine the codebase to ensure optimal performance and adherence to coding standards.</li> <li>● Create a final Docker image containing all necessary components (web app, scripts, dependencies) for deployment.</li> <li>● Carry out the testing phase and record test analysis.</li> </ul>
July 20 - July 26 (1 week)	<ul style="list-style-type: none"> <li>● Create a comprehensive project delivery package containing documentation, the final Docker image, and a Readme file.</li> </ul>

## Commitments

I plan to spend 6 hours per day, so roughly around 35 hours per week(mon-sat). On Sunday, I would like to spend time with my family. I have a summer break for a significant amount of time, so no university commitments. In any case, I assure the organization that I can dedicate the above-mentioned hours (at least 35 hours per week).