filez++ (SSH File Manager)

A PROJECT REPORT
submitted by

RAHUL R
LKTE18MCA065

to

the APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the degree

of

Master of Computer Applications

Department of Computer Applications
RAJIV GANDHI INSTITUTE OF TECHNOLOGY
(Government Engineering College)
KOTTAYAM - 686 501, KERALA

June 2021

DECLARATION

I undersigned hereby declare that the project report ”filez++ (SSH File Manager)” , submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Name of supervisor(s). This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

PAMPADY

June 12, 2021 RAHUL R

DEPARTMENT OF COMPUTER APPLICATIONS

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

KOTTAYAM

CERTIFICATE

This is to certify that the report entitled **'filez++ (SSH File Manager)'** submitted by **'Mr. RAHUL R'** to The APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of **Master of Computer Applications** is a bonafide record of the project work carried out by him/her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor **External Supervisor**

External Examiner **HEAD OF THE DEPARTMENT**

ACKNOWLEDGEMENT

The project **filez++ (SSH File Manager)** wouldn't have been a success if it wasn't for the support and guidance from various sources throughout the development. I use this opportunity to show gratitude towards everyone who have had impact on the project directly or indirectly.

I express my sincere thanks to **Dr. Jalaja M.J**, Principal, Rajiv Gandhi Institute of Technology for providing the ambiance for the completion of my project.

I also express my gratitude to **Prof. John C John**, H.O.D of Department of Computer Applications for providing us with adequate facilities, ways and means by which we were able to complete this project.

I would like to thank my staff advisor and project co-ordinator **Prof. Shalu Murali**, Assistant Professor(Adhoc) at Department of Computer Application.

I would like to thank my project guide Assistant Professor **Jane George**, Professor at Department of Computer Applications for her support and guided me in the project. I take this opportunity to thank all the technical staffs of Department of Computer Application for their help.

I also express my gratitude to **SunTec Digital Solutions** for providing me with adequate facilities, Support and guidance ways and means to complete this internship.

Last but not the least I place a deep sense of gratitude to our family members and our friends who have been constant source of inspiration during the project.

ABSTRACT

The Project is to build a Utility Tool for Browsing files from a Remote server using protocols like Secure Shell Host (SSH), Secure File Transfer Protocol (SFTP). The Software will be developed as an installable Desktop Application using Electron.JS (Packaging Tool) by using Web Technologies including Node.JS (Runtime Environment for JavaScript, Server Side), Express.JS (MVC Framework), Angular (Front End Framework). The outcome will be an installable package, that is supported by many Operating Systems like Windows, Ubuntu, etc. Target Users are mostly Software Developers who use remote servers. Existing Softwares are available, but have drawbacks. This project mainly focuses on fixing those issues and building a reliable GUI File Manager for both Local and Remote File Systems.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Need for the Project

As the current pandemic is going on, most employees are working from home remotely, So there is an increased need for using a file manager that can access remote files of a company, a college etc. Even though remote accessing of files can be done through command prompt, naive users will find it difficult to use and to educate them it is hard. There are some File managers that can access remote servers, they have a lack of user experience and the UI is very old, that most naive users find it difficult to use. And most of them require Server Side Installation along with Client side application. More than that filez++ will be developed as a desktop application, that can access both local and remote files through protocols like SSH, SFTP etc.

1.2 Outline of the Report

Requirement analysis, Specification, Hardware Requirements and Software Requirements required for Development and Deployment are shown in Chapter 2. System Architecture and Features of the Software is shown in Chapter 3. Data Model Used for sending and receiving commands and operations and Screenshots are shown in Chapter 4. Real time System Testing details are shown in Chapter 5. Application Packaging for different Operating Systems and Distribution are given in Chapter 6. Report is concluded in Chapter 7.

1.3 Motivation

This Project is inspired by filsZilla, a Software used for accessing Remote Files using protocols like SSH, SFTP, etc.

1.4 Scope of the Project

The Software can be used in all major Operating Systems like Windows, Ubuntu, MAC, (Both 32 and 64 bit architecture) etc. Anyone who have access to a remote server can use the SSH feature and others can only use the local file system accessing feature. Although the main users will be Software Engineers who have to use remote servers, Web Developers for accessing hosted Servers, College Professors and Students for Educational Purposes.

CHAPTER 2

REQIUREMENT ANALYSIS AND SPECIFICATON

2.1 System Study

Requirement Analysis was done by collecting the features of existing Similar Softwares as well as the mandatory features of a GUI File Manger like the Windows Explorer. Features like add connection, test connection, delete files, rename files, copy files etc. were found required for the basic functionality of the software

2.1.1 Existing System

There are many similar existing systems, but this project was particularly inspired by a software called fileZilla, which is a SSH file manager. It have a few limitations like:

- Cannot save connections
- Cannot connect to multiple ssh servers at once
- User Experience Issue, hard to use for naive users
- Server-side installation required
- Not Portable (User have to install the software in their PC)

2.1.2 Proposed System

This software is primarily built for improving User Experience. Features and improvements from Existing Systems are

- Add/Edit/Test/Delete Connections
- Connect to multiple SSH servers at once.
- GUI is easy to use
- Server-side installation is not required
- Portable (Can run without installation on Users PC)
- Can run on Windows, Ubuntu, Fedora, Debian, MAC, etc.
- Does not take much resources like RAM and CPU.

2.2 System Specification

2.2.1 Hardware Specification

- RAM : Greater than 1 GB, 2GB Recommended
- Storage : 350 MB free space, 400 MB Recommended (Executables and packages size is approximately 296 MB and cache, temp files and local storage might also take up some space)

2.2.2 Software Specification

For Development used Node.js, Electron.js, Express.js, Angular and Atom was the Editor used. For running the Application there is no requirement, any desktop operating system can be used and there is no need to install any other software or plugins to run the app. No need to install either, the App is portable.

2.2.3 Software Tools

For development of this software Web Technologies are used and are packaged into Executables. Programming Languages, Frameworks and Tools used are Node.js [1] which is a single threaded asynchronous runtime for JavaScript [2], Express.js [3] which is an MVC Framework for Node.js, Angular [4] which is the component based framework for JavaScript and Electron.js [5] is used for packaging it into executable files like .exe, .bin etc. SSH client package used is SSH2-Promise [6] an asynchronous wrapper of SSH2 [7] package. For Development Atom IDE [8] is used. WebSockets [9] are also used for the communication of commands, operations, status, progress etc.

CHAPTER 3

SYSTEM MODELING

3.1 System Architecture

First Angular source is compiled and built for production, then the Node.js source is embedded into electron and is compiled and built for packaging and builds for each operating system is obtained and distributed. Node runs as a runtime instance on Electron along with embedded chromium web view.

The Main Process is the first one getting started, which is an instance of Electron.js and Node.js, It then loads the Browser Window which is an instance of Chromium, the angular build files are loaded and the Renderer Process will will be initiated which can be used to interact with Browser Window and Main Process. Browser Window will not have access to local files or any dependencies or packages added in Main Process. To access it, remote module from Renderer Process is used.

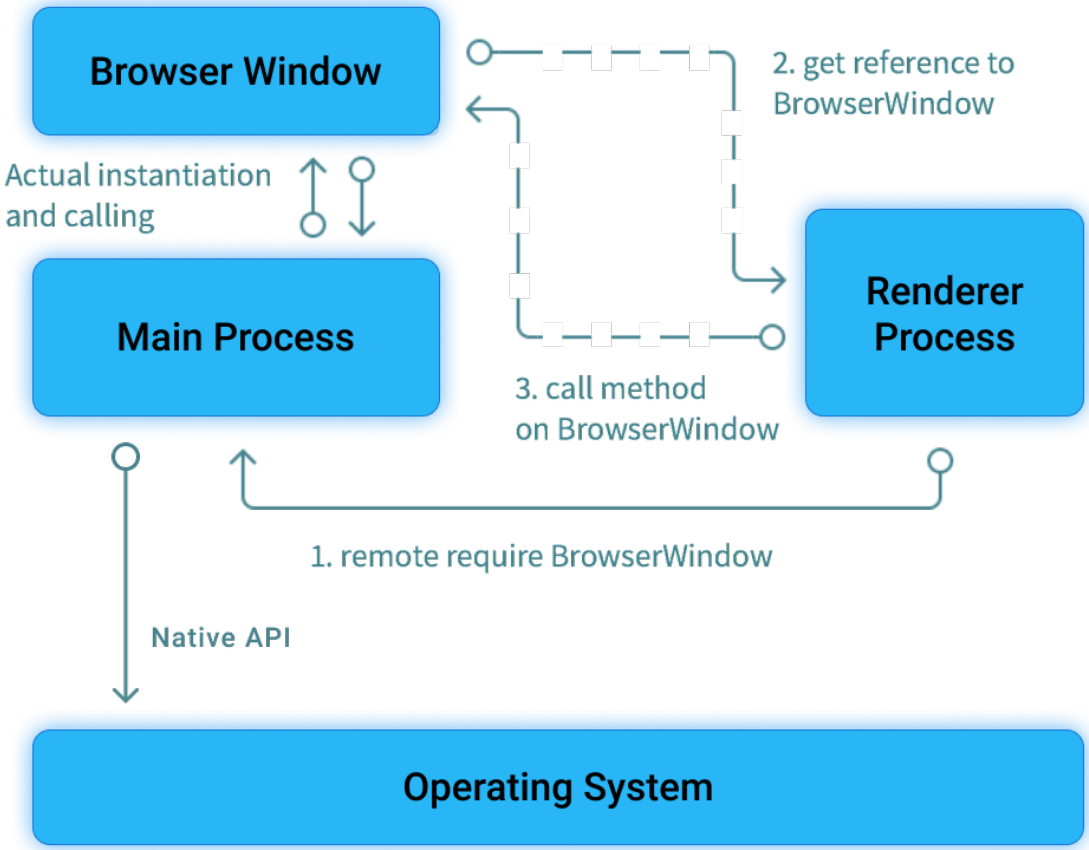

Fig. 3.1. System Architecture

3.2 Features

These are some of the features in the software

3.2.1 Add Connection

User can add an SSH connection by entering the details, which will be stored permanently for connecting to Server.

3.2.2 Test Connection

Check if the connection is working

3.2.3 Edit Connection

Modify connection

3.2.4 Delete Connection

Remove Connection

3.2.5 View Directory Contents

User can view both SSH and local files and folders and can see if the sub folders are filled (have content) or not, or is readable

3.2.6 Create New File or Folder

User can create new file or folder

3.2.7 Rename

User can rename file or folder

3.2.8 Delete

User can delete multiple files or folders (Permanent, no recycle bin options)

3.2.9 Cut, Copy, Paste

User can copy or move and paste multiple files or folders from and to any ssh or local file system. Implemented by creating read stream from source and creating a write stream in destination.

3.2.10 Shortcuts

Shortcuts as well as other GUI methods can be used to use the above features

3.2.11 Progress

These operations are performed in the background and the status and progress can be viewed in the queue. Users will get in-app notification when a task is completed or failed

3.2.12 Search

Search items in a directory

CHAPTER 4

SYSTEM DESIGN

4.1 Introduction

Two data passing methods are used to transmit data between node.js and angular instances, one of them is to use traditional http protocol, Where the application first reserves an open TCP port from Operatiing system and listens to that port. this port is sent to angular using electron's ipcMain and ipcRenderer modules. Once the port is received angular will connect to node.js in that port and initiates a WebSocket connection. Which will be the primary medium of sending operation requests, getting status, progress etc, except for directorty listing, which uses the http protocol, due to it's synchronous nature.

4.2 Data Model Design

Some examples of request and response of web sockets, data is sent as JSON in a stringified form

4.2.1 File Folder delete - request model

```
{
  "process_id" : number, // uniquely generated
  "type": "delete",
  "source": {
    "server": String, // "user@host_ip"
    "baseFolder": String // path to base folder
```

```
  },
  "files": String[] // filenames of all selected items
}
```

4.2.2 File Folder copy - request model

```
{
  "process_id" : number,
  "type": "copy",
  "source": {
    "server": String, // "user@host_ip"
    "baseFolder": String // path to base folder
  },
  "target": {
    "server": String, // "user@host_ip"
    "baseFolder": String // path to base folder
  },
  "files": String[] // filenames of all selected items
}
```

4.2.3 operation success - response model

```
{
  "process_id" : number,
  "status" : "completed"
}
```

4.2.4 operation failure - response model

```
{
  "process_id" : number,
  "status" : "failed",
  "error_message" : message, // for user to see
  "dev_log" : error_data // for developer
}
```

4.3 UI Designing

4.3.1 Screenshots

No templates or libraries are used to design UI, Used builtin Grid and Flex System of CSS.

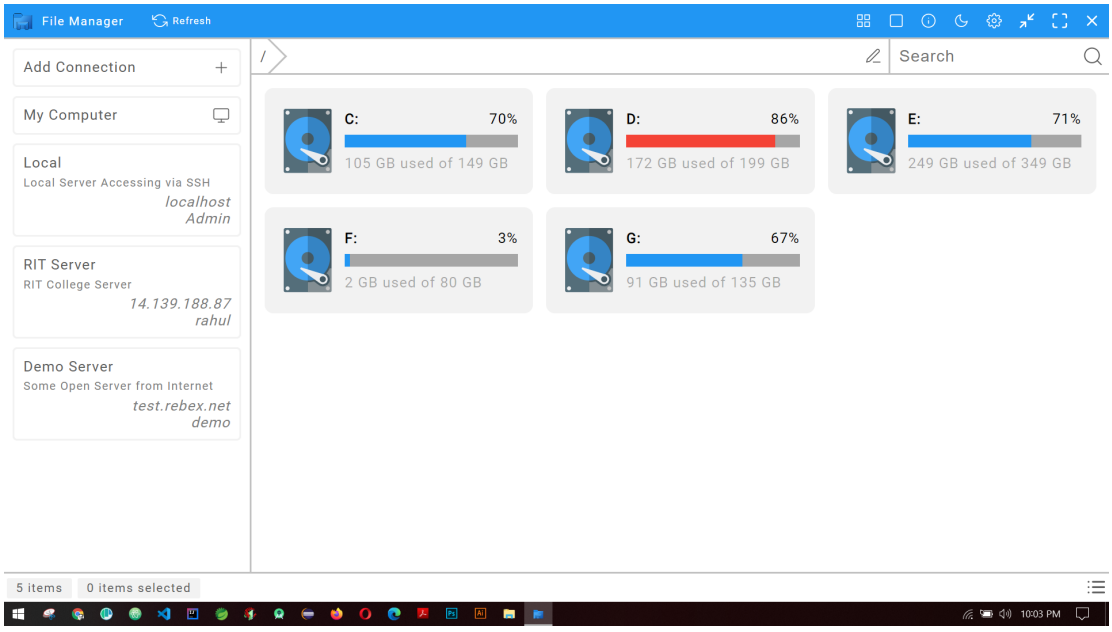

Fig. 4.1. Local Drives

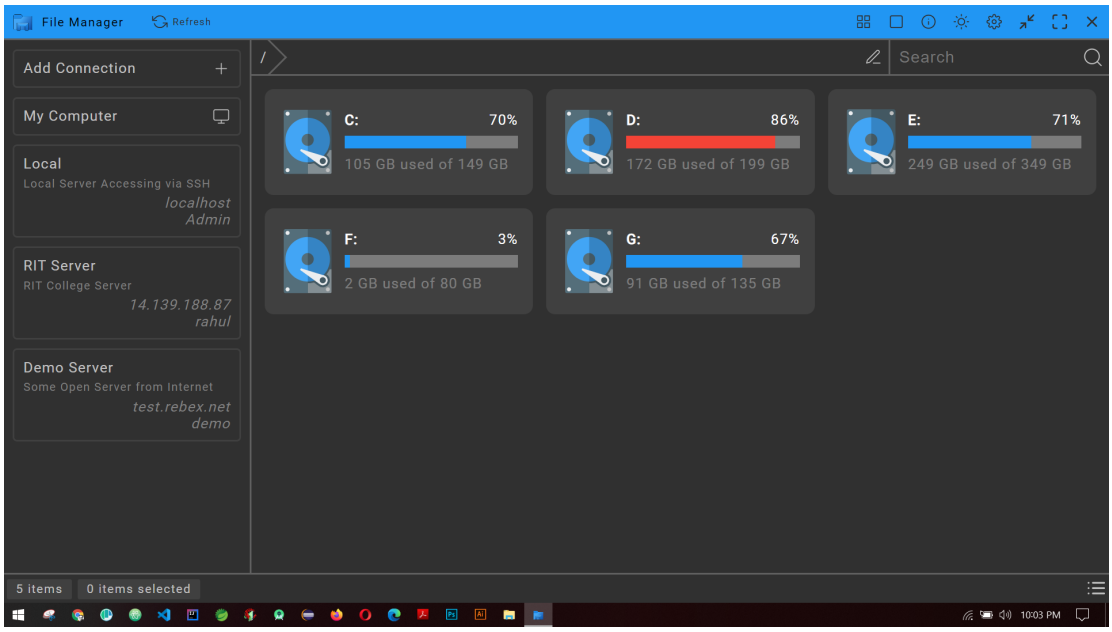

Fig. 4.2. Dark Mode

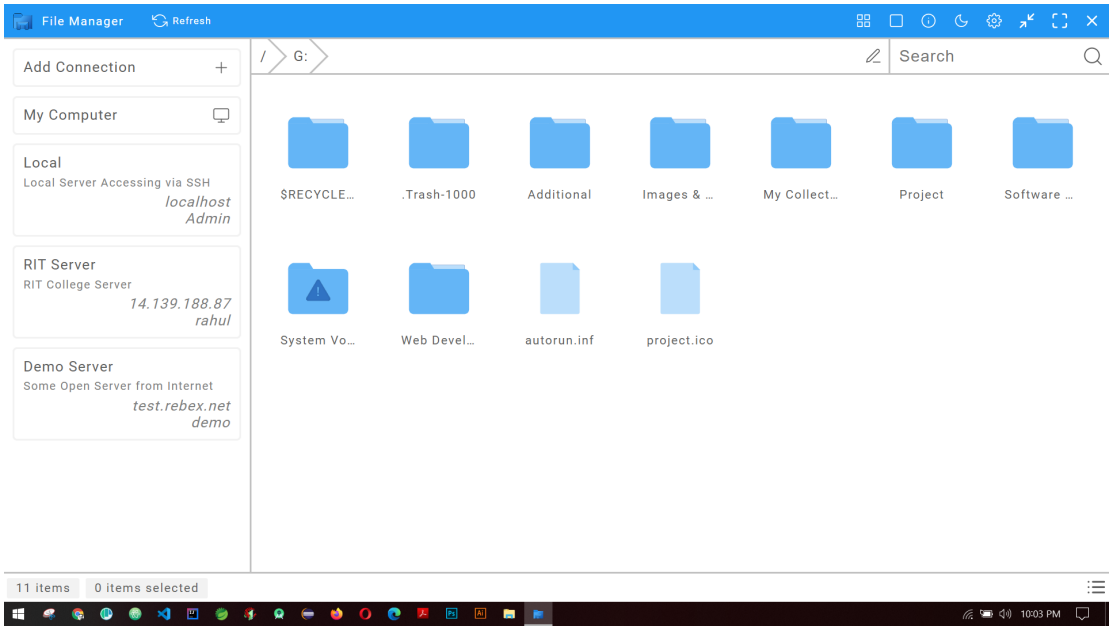

Fig. 4.3. List Directory Contents

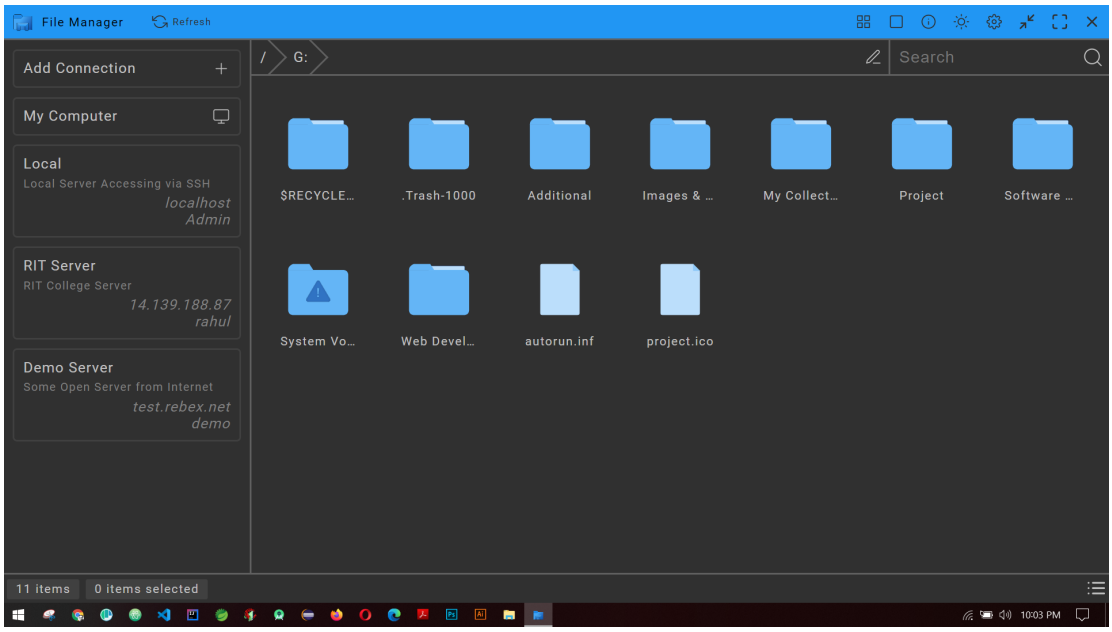

Fig. 4.4. List Directory Contents Dark Mode

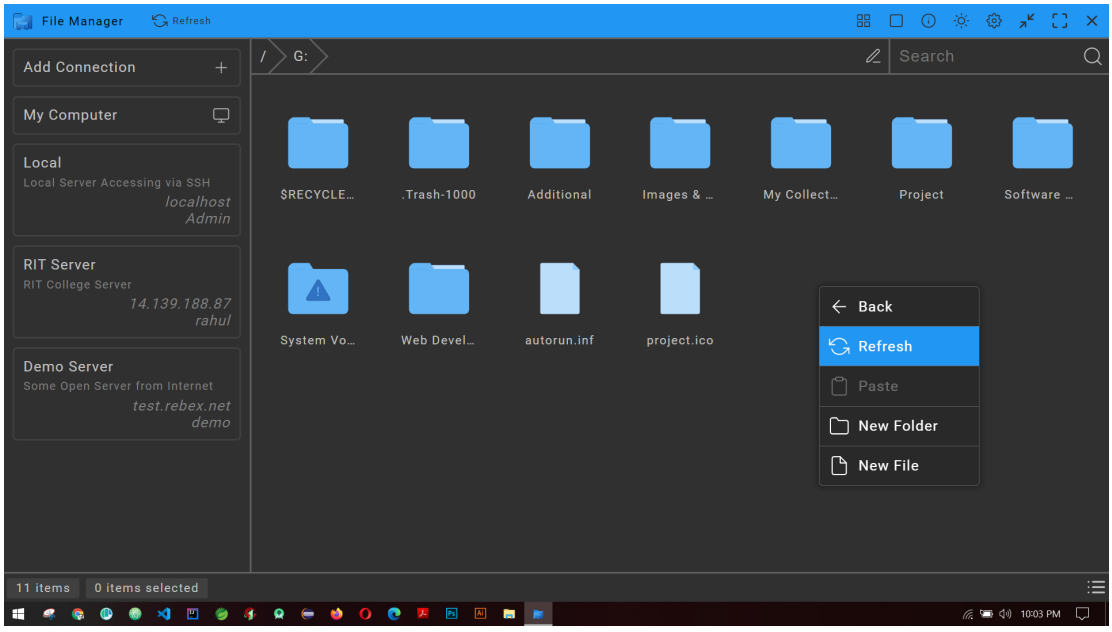

Fig. 4.5. Right Click Options

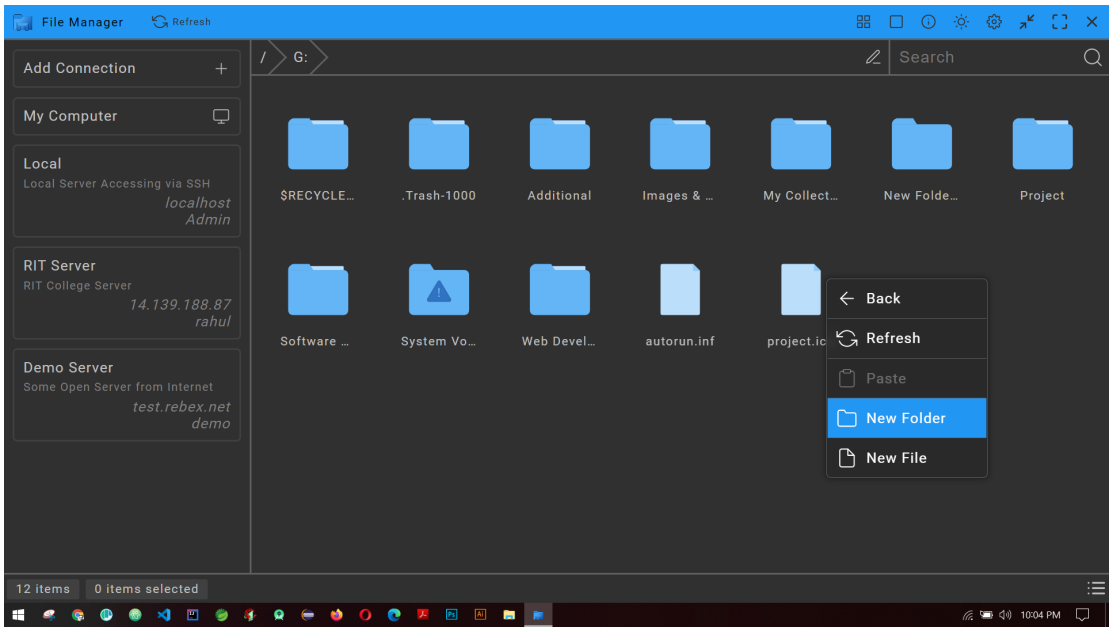

Fig. 4.6. Create New Folder

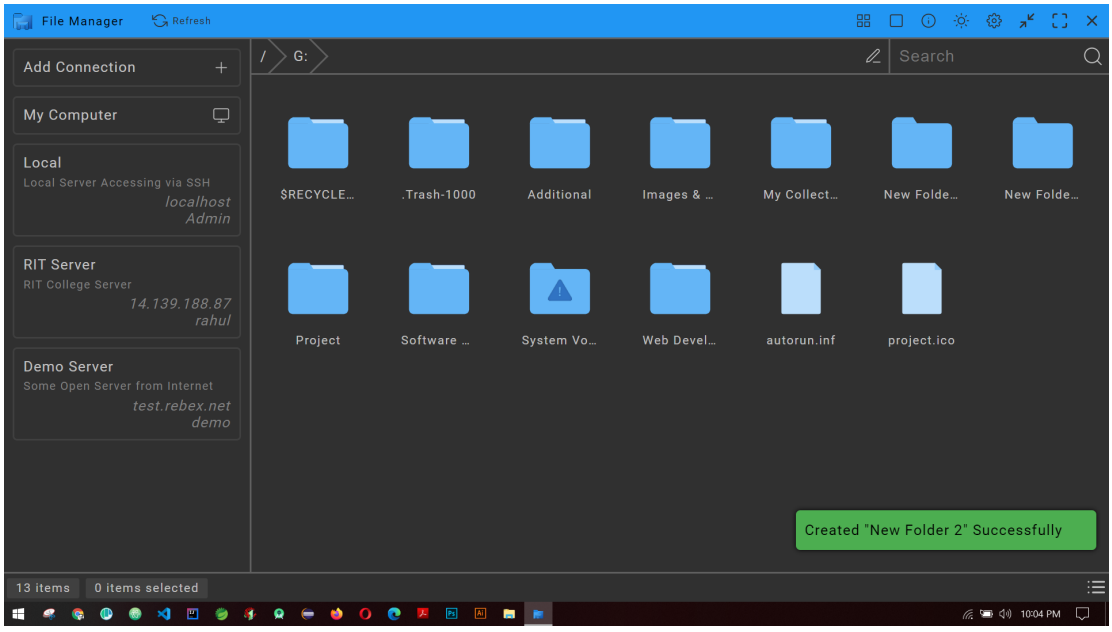

Fig. 4.7. Success Notification

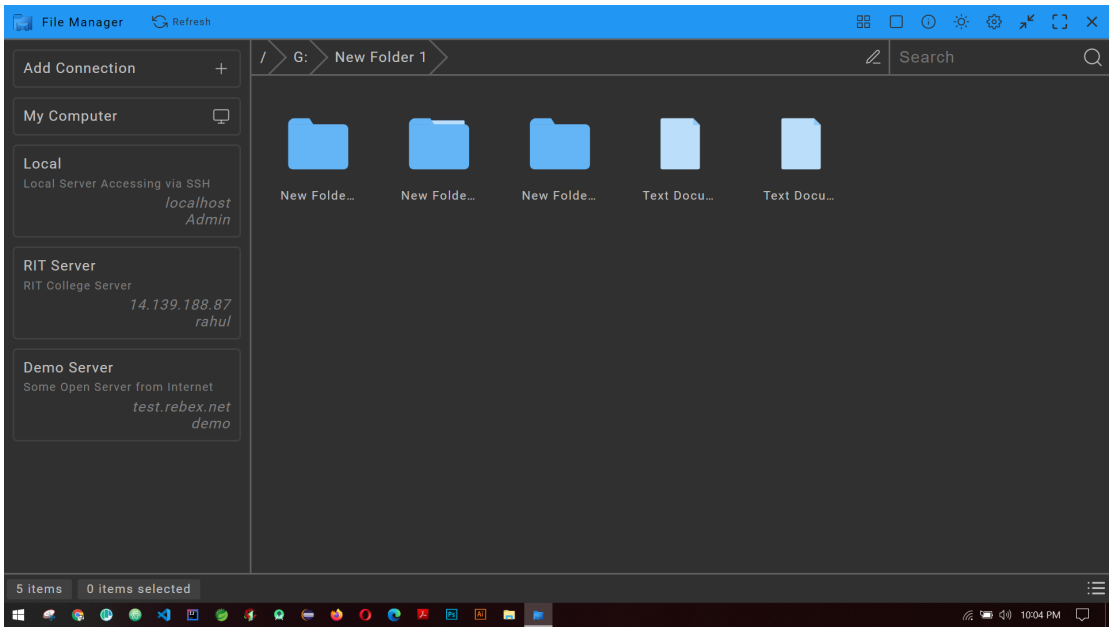

Fig. 4.8. New Folder Created

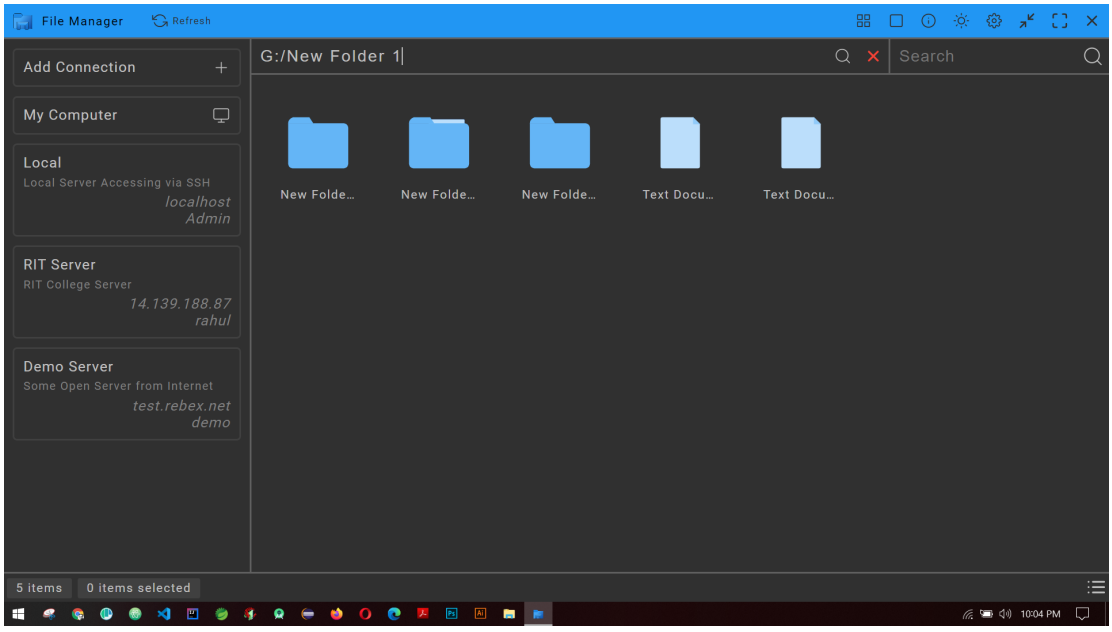

Fig. 4.9. Edit Path

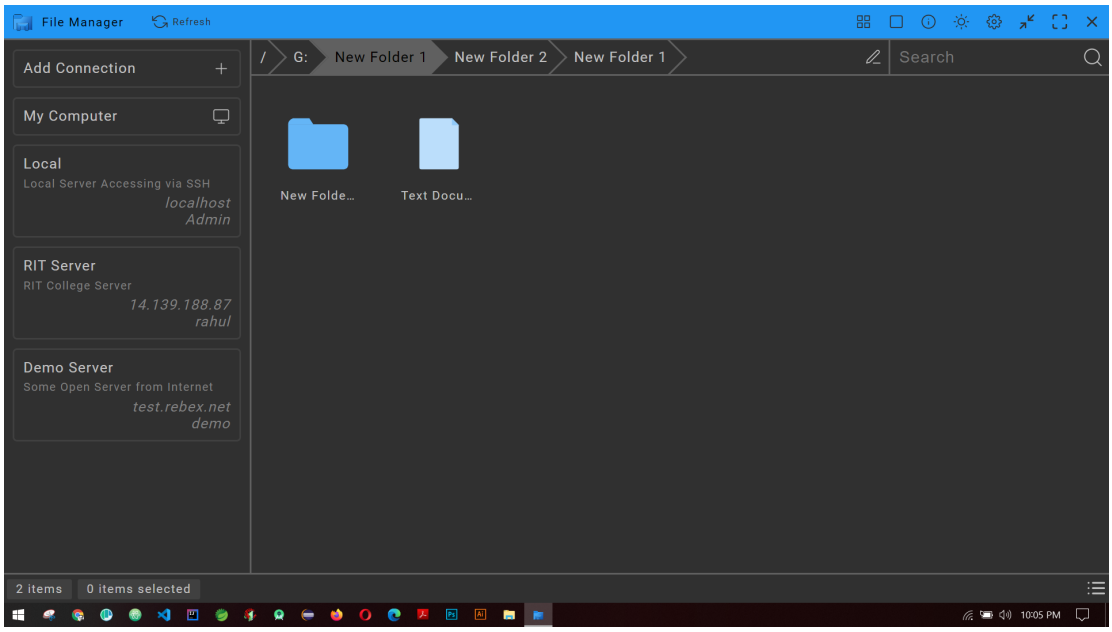

Fig. 4.10. Move to Path

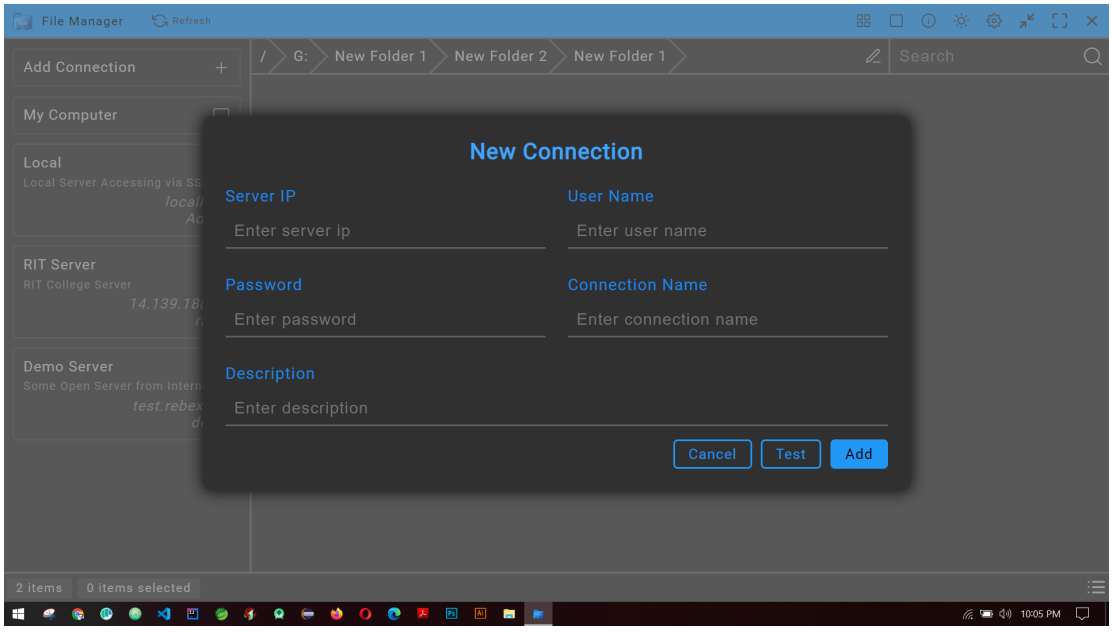

Fig. 4.11. Add Connection

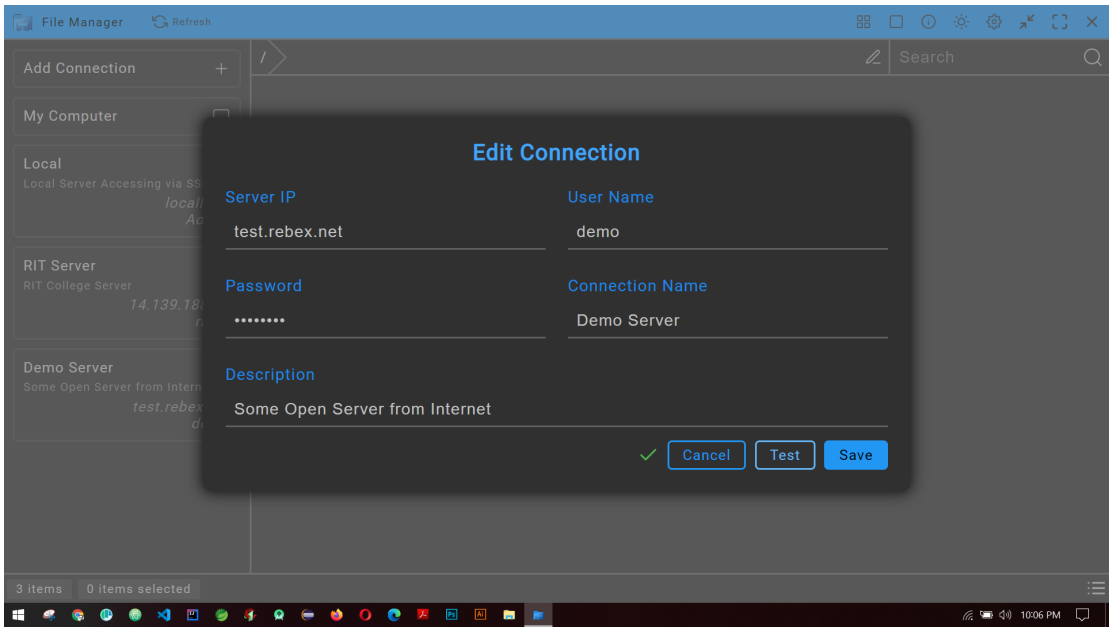

Fig. 4.12. Edit Connection

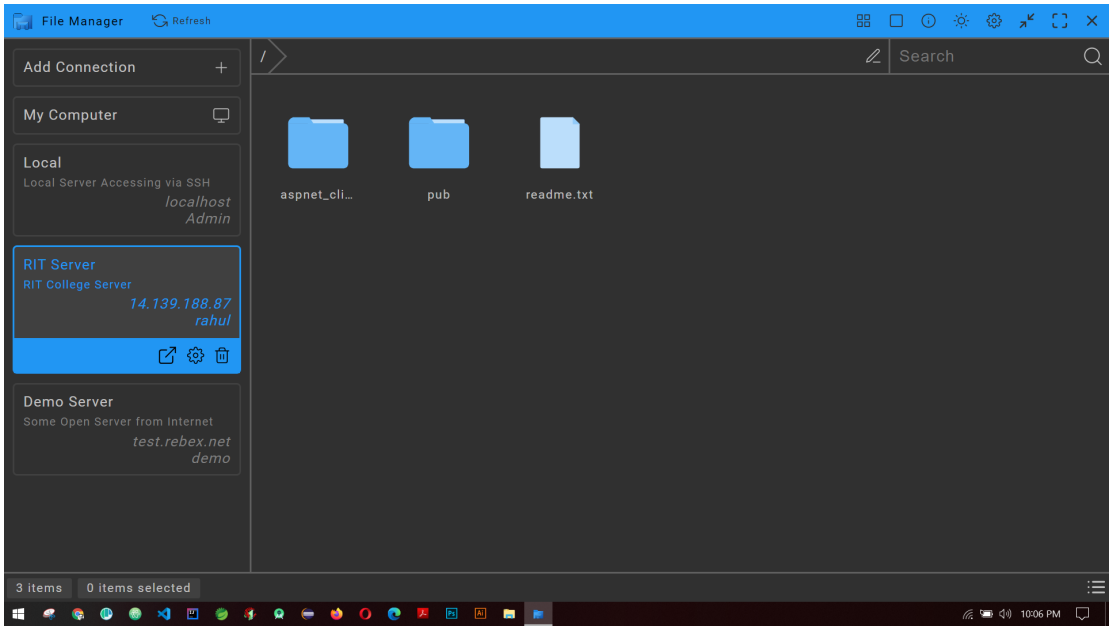

Fig. 4.13. Connect to SSH

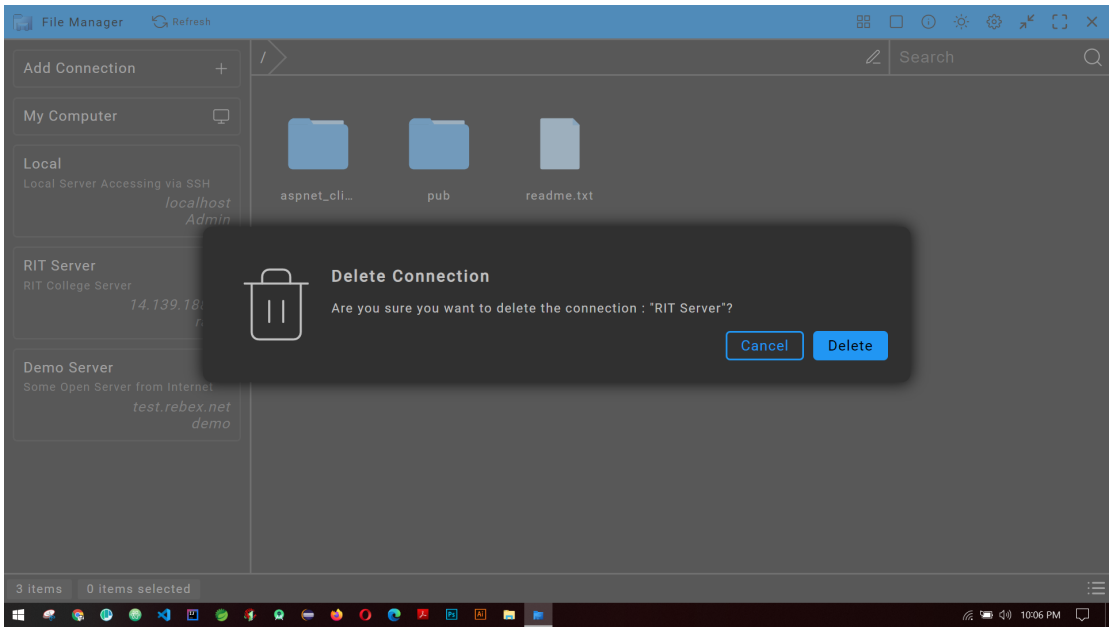

Fig. 4.14. Delete Connection

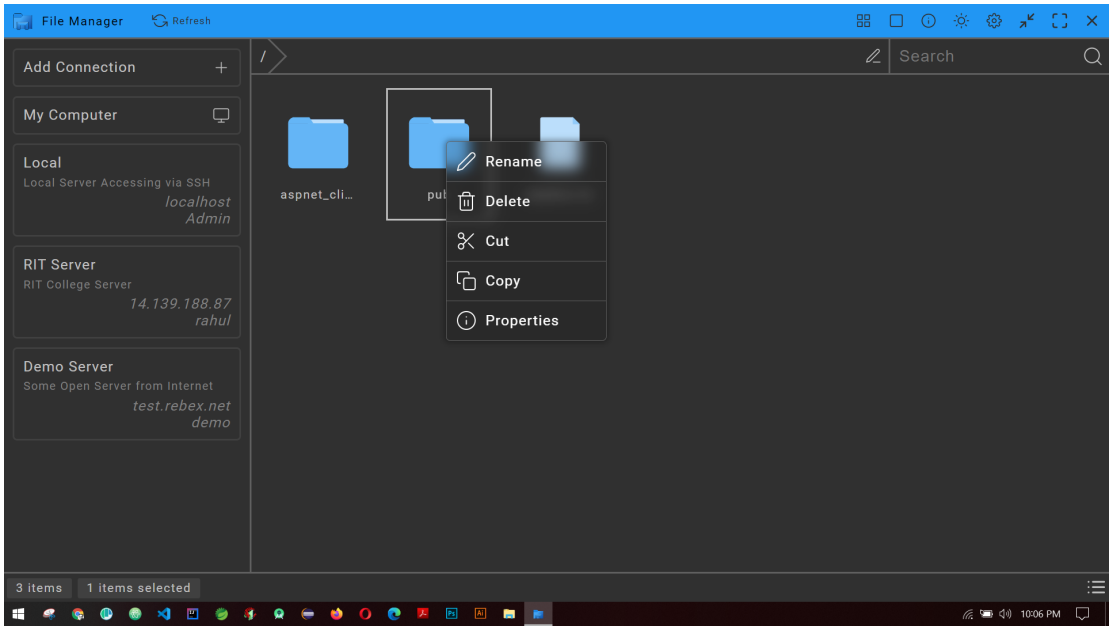

Fig. 4.15. Folder Right Click

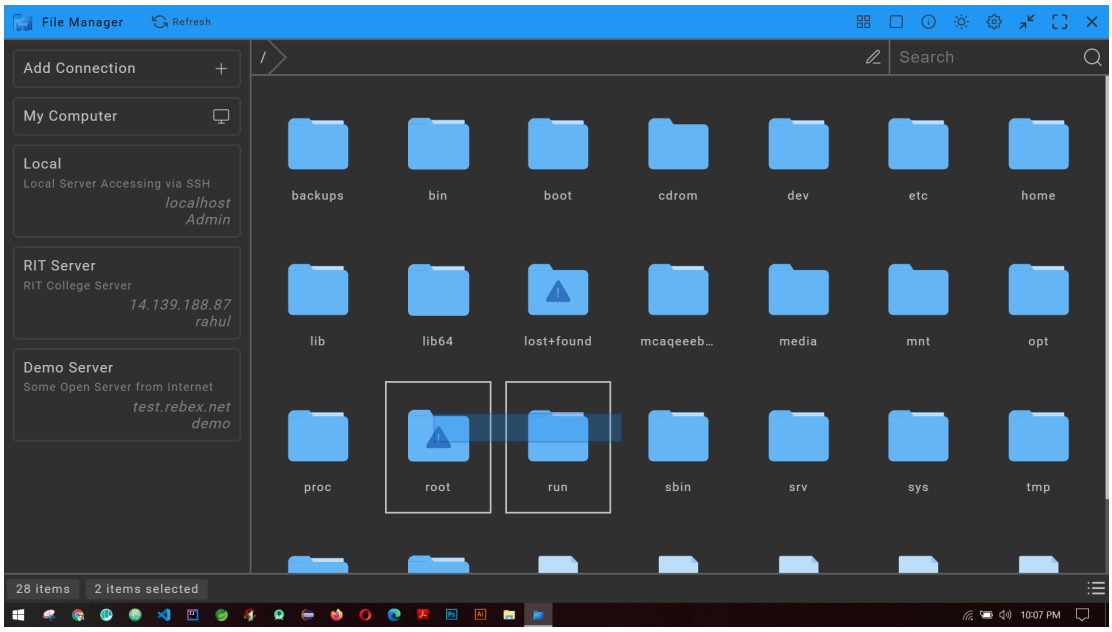

Fig. 4.16. Drag Select Files and Folders 1

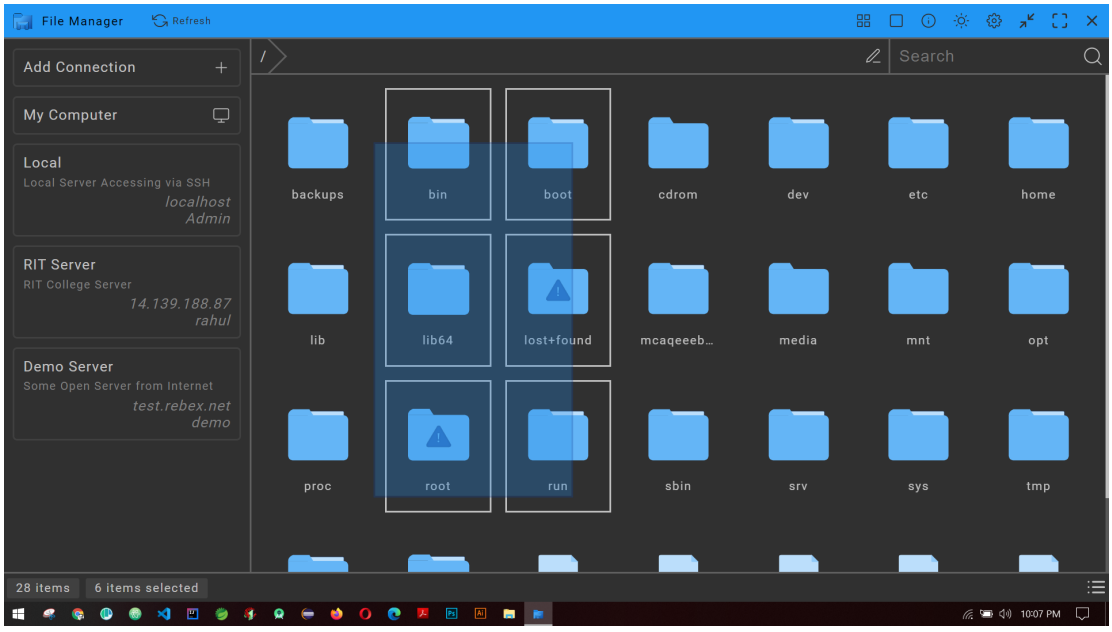

Fig. 4.17. Drag Select Files and Folders 2

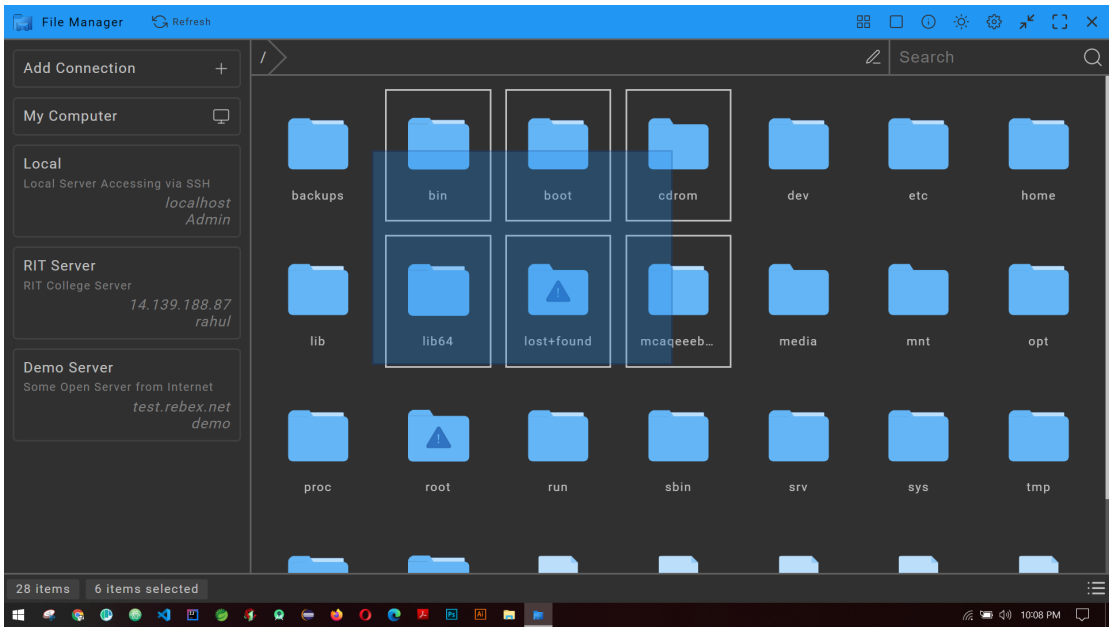

Fig. 4.18. Drag Select Files and Folders 3

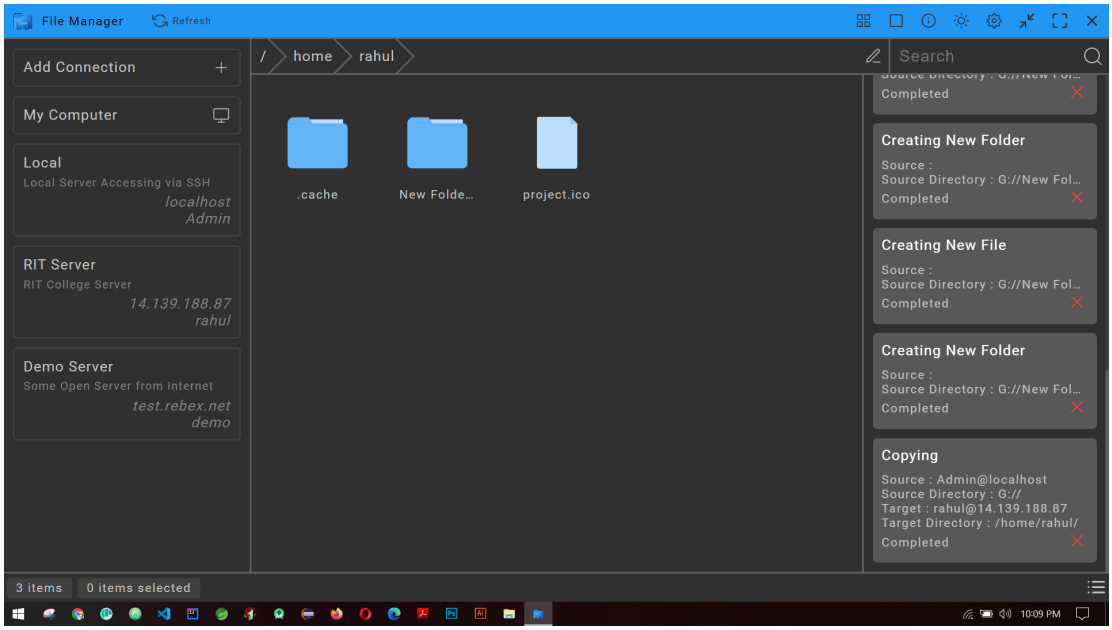

Fig. 4.19. Background Task Queue

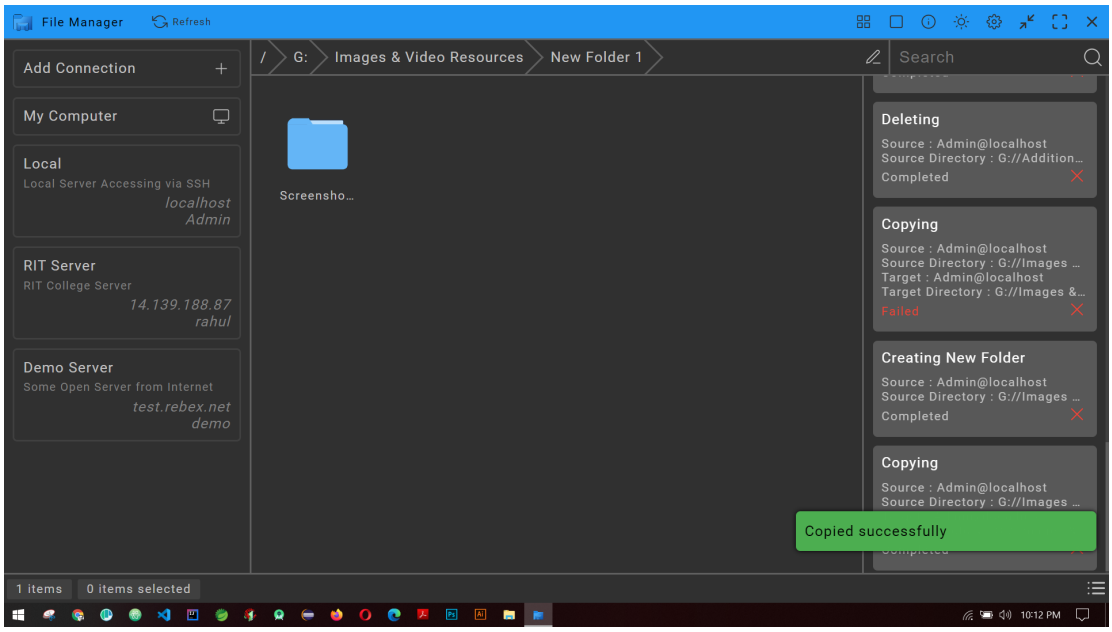

Fig. 4.20. View Old Operations

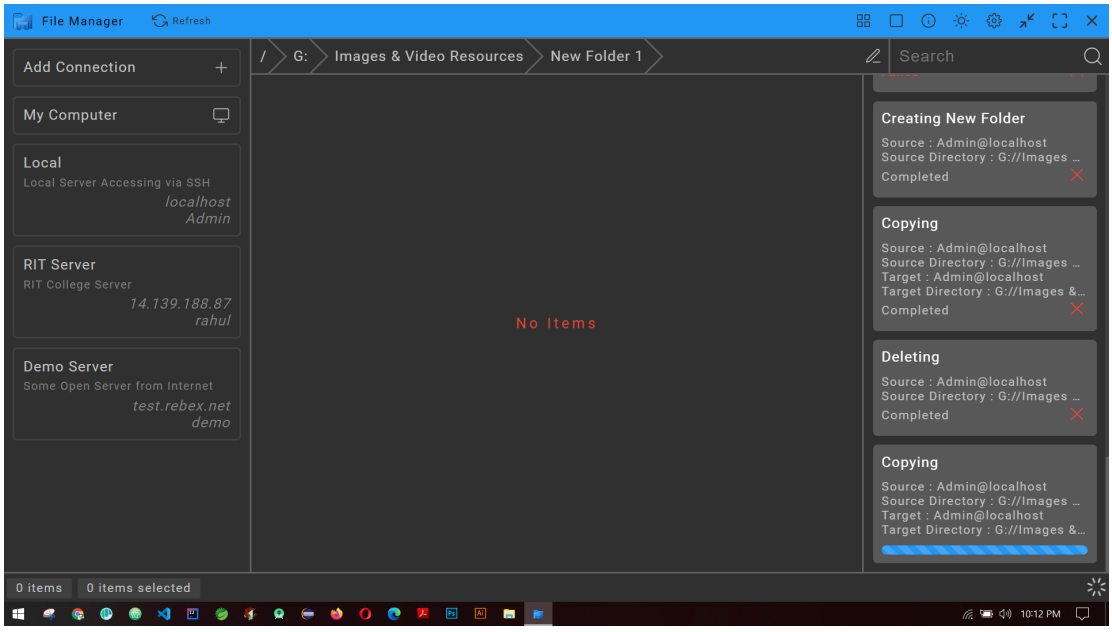

Fig. 4.21. Background Task in Progress

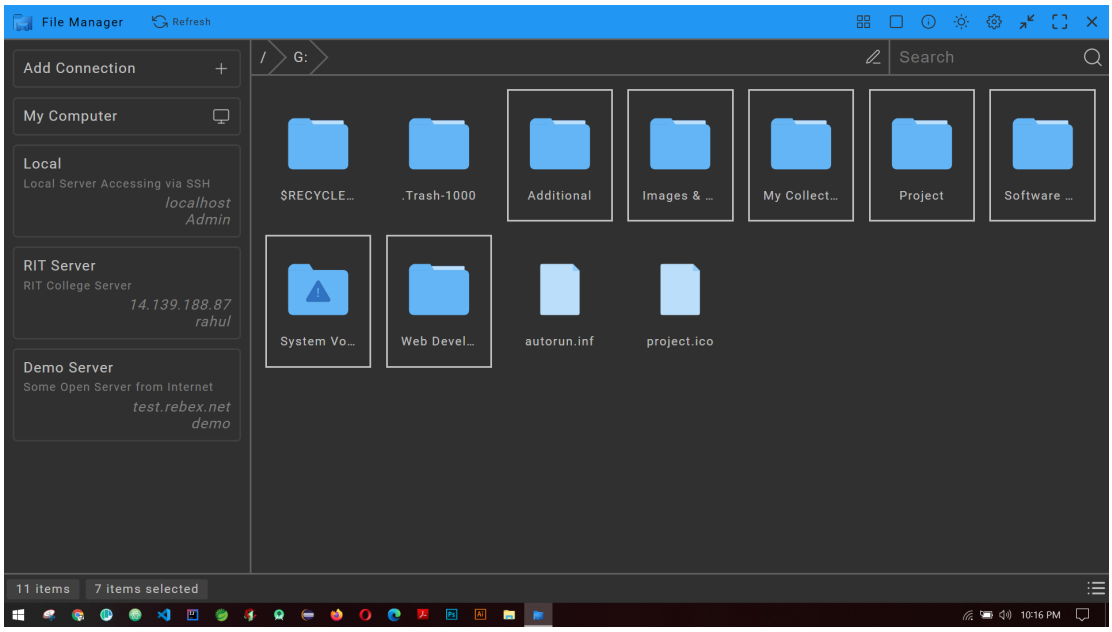

Fig. 4.22. Shift + Select

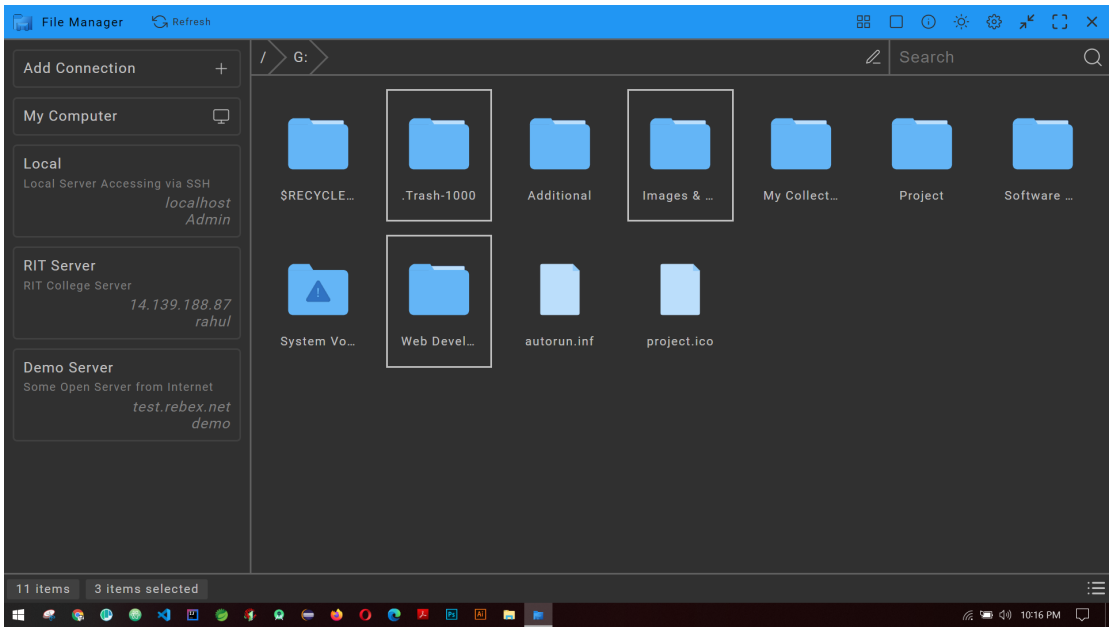

Fig. 4.23. Ctrl + Select

CHAPTER 5

SYSTEM TESTING

- Realtime error prevention methods are used by implementing data type system using TypeScript [10].
- Other plugins used for testing are ES Lint [11] and TS Lint [12],
- Some important functions are tested as part of unit Testing
- Built in assert modules were also used for testing.
- SSH connections are tested for Windows and Linux Servers.

CHAPTER 6

SYSTEM IMPLEMENTATION

These packages can be distributed and can be used without any other installations or dependencies. It uses the built in localStorage for storing configurations, settings, etc Which will be stored as AppData, rwmp files etc. The Software is portable, customizable and can be generated for the following types:

- Windows x32
- Windows x64
- Ubuntu x32
- Ubuntu x64
- MAC x64

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

The executable files generated are supported by X-OS and can be distributed without needing to install any server side software, Neither client side installation is not required, as the software is portable. Once the specific version is built, it can be packaged and distributed to users. And they can use it as a regular Application. As the application does not require any hosting, application can be directly sent to users or, can be uploaded to any sites for distribution. And since there are no restrictions or limitations for using the app, any user can easily download and use the application.

REFERENCES

[1] Node.js, “https://nodejs.org/en/docs/.”

[2] JavaScript, “https://developer.mozilla.org/en-us/docs/web/javascript.”

[3] Express.js, “https://expressjs.com/.”

[4] Angular, “https://angular.io/.”

[5] Electron.js, “https://www.electronjs.org/docs.”

[6] SSH2-Promise, “https://www.npmjs.com/package/ssh2-promise.”

[7] SSH2, “https://www.npmjs.com/package/ssh2.”

[8] Atom IDE, “https://atom.io/.”

[9] WebSocket, “https://www.npmjs.com/package/ws.”

[10] TypeScript, “https://www.typescriptlang.org/docs/.”

[11] ES Lint, “https://eslint.org/.”

[12] TS Lint, “https://palantir.github.io/tslint/.”