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

July 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 guide Prof. Jane George, Assistant Professor, Department of Computer Applications. 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
July 8, 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' (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** is a bonafide record of the project work carried out by him 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**

SUN/HRD/2021/06-25/046
25-June-21

TO WHOMSOEVER IT MAY CONCERN

This is to certify that**. Mr. Rahul R (T30011)**, student of Rajiv Gandhi Institute of Technology, Kottayam has successfully completed his project internship with us. He has been associated with us as an "Intern" from 03 March 2021 for a duration of six months.

The title of his project is "filez++ (SSH File Manager".

We found Rahul R to be sincere, hardworking and dedicated to the tasks assigned.

We wish him all success in his future endeavors!

This certificate has been issued as per his request, without any further commitment on the part of the company.

for **SunTec Digital Solutions Private Limited,**

Prakash P Nair
Head – Human Resources

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 Applications.

I would like to thank my project guide **Prof. Jane George**, Assistant Professor, 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 Applications 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

Acknowledgement i

Abstract ii

List of Figures vi

CHAPTER 1. Introduction 1

1.1 Need for the Project 1

1.2 Outline of the Report 1

1.3 Motivation 2

1.4 Scope of the Project 2

CHAPTER 2. REQIUREMENT ANALYSIS AND SPECIFICATON 3

2.1 System Study 3

2.1.1 Existing System 3

2.1.2 Proposed System 4

2.2 System Specification 4

2.2.1 Hardware Specification 4

2.2.2 Software Specification 4

2.2.3 Software Tools 5

2.2.4 Main Features of Node.js 6

2.2.5 Angular 7

2.2.6 Electron.js 8

CHAPTER 3. SYSTEM MODELING **9**
3.1 Features . . . 9
3.1.1 Add Connection . . . 9
3.1.2 Test Connection . . . 9
3.1.3 Edit Connection . . . 9
3.1.4 Delete Connection . . . 10
3.1.5 View Directory Contents . . . 10
3.1.6 Create New File or Folder . . . 10
3.1.7 Rename . . . 10
3.1.8 Delete . . . 10
3.1.9 Cut, Copy, Paste . . . 10
3.1.10 Shortcuts . . . 11
3.1.11 Progress . . . 11
3.1.12 Search . . . 11
3.1.13 Open File . . . 11
3.1.14 Open Shortcut File . . . 11
3.2 System Architecture . . . 12

CHAPTER 4. SYSTEM DESIGN **13**
4.1 Introduction . . . 13
4.2 Data Model Design . . . 13
4.2.1 File Folder delete - request model . . . 13
4.2.2 File Folder copy - request model . . . 14
4.2.3 Operation failure - response model . . . 14
4.2.4 Operation partially succeeded - response model . . . 14
4.2.5 Operation success - response model . . . 15
4.2.6 Progress - response model . . . 15
4.3 Scrum Details . . . 15
4.3.1 Product Backlog . . . 15
4.3.2 Scrum Backlog . . . 16
4.4 UI Designing . . . 16
4.4.1 Screenshots . . . 17

CHAPTER 5. SYSTEM TESTING **29**

CHAPTER 6. SYSTEM IMPLEMENTATION **31**

CHAPTER 7. CONCLUSION AND FUTURE SCOPE **32**

References **33**

LIST OF FIGURES

3.1 System Architecture 12

4.1 Product backlog 15
4.2 Sprint backlog 16
4.3 Local Drives 17
4.4 Dark Mode 17
4.5 List Directory Contents 18
4.6 List Directory Contents Dark Mode 18
4.7 Right-Click Options 19
4.8 Create New Folder 19
4.9 Success Notification 20
4.10 New Folder Created 20
4.11 Edit Path 21
4.12 Move to Path 21
4.13 Add Connection 22
4.14 Edit Connection 22
4.15 Connect to SSH 23
4.16 Delete Connection 23
4.17 Folder Right Click 24
4.18 Drag Select Files and Folders 1 24
4.19 Drag Select Files and Folders 2 25
4.20 Drag Select Files and Folders 3 25
4.21 Background Task Queue 26
4.22 View Old Operations 26
4.23 Background Task in Progress 27
4.24 Shift + Select 27
4.25 Ctrl + Select 28

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 an 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 the technologies used are:

- Node.js

- Electron.js
- Express.js
- Angular
- IDE: Atom

Main dependencies used are:

- WebSockets
- SSH-Promise-2

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 runtime for JavaScript [2] which is high-level, single-threaded, garbage-collected, interpreted (or just-in-time compiled), prototype-based, multi-paradigm, dynamic language with a non-blocking event loop and is built using the JavaScript Engine of Chrome named "V8", Express.js [3] which is an MVC Framework for Node.js, Angular [4] which is the component based framework for JavaScript and TypeScript 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.

2.2.4 Main Features of Node.js

- **High-level** : High-Level refers to the abstraction the language provides over the machine's bare-metal hardware. JavaScript is considered high-level because it does not require direct interaction with the operating system, hardware. In addition, it does not require memory-management like C/C++ because the runtime always uses garbage-collection.

- **Single-threaded** : This means it has one call stack and one memory heap. As expected, it executes code in order and must finish executing a piece code before moving onto the next. Single-threaded means that JS can only run one instruction at a time, even if CPU has multiple cores and available threads. This property can be modified by deploying multiple instances of Node.js by using load balancing tools.

- **Garbage collection** : Garbage collection is the process that does automatic memory management. The purpose of a garbage collector is to monitor memory allocation and determine when a block of allocated memory is no longer needed and reclaim it. This automatic process is an approximation since the general problem of determining whether or not a specific piece of memory is still needed is undecidable.

- **Interpreted or Just-in-Time Compiled** : Interpreted means the source code is converted to bytecode and executed at runtime (as opposed to being compiled to a machine code binary at build time). This is also why JS is commonly called a "scripting language". Originally, it was only interpreted, but modern JS engines like V8, Spidermonkey, and Nitro use various techniques to perform Just-in-Time Compilation or JIT for better performance. Developers still use JS like an inter-

preted language, while the engine magically compiles parts of source code to low-level machine code behind the scenes.

- **Prototype Inheritance** : Prototype Inheritance means that objects can inherit behaviors from other objects. This differs from classical inheritance where you define a class or blueprint for each object and instantiate it. Unlike java where you need to create a new class to extend it's features (Inheritance), it is possible to inherit/extend a class in JavaScript by modifying the prototype.

- **Multi-Paradigm** : Multi-Paradigm means the language is general-purpose or flexible. JS can be used for declarative (functional) or imperative (object-oriented) programming styles.

- **Dynamic Weakly Typed** : Dynamic most often refers to the type system. JS is dynamic weakly typed language, meaning developer do not annotate variables with types (string, int, etc) and the true types are not known until runtime.

- **Non-blocking event loop** : Event Loop refers to a feature implemented by engines like V8 that allow JS to offload tasks to separate threads. Browser and Node APIs execute long-running tasks separately from the the main JS thread, then enqueue a callback function (which you define) to run on the main thread when the task is complete. This is why JS is called non-blocking because it should only ever wait for synchronous code from your JS functions. Think of the Event Loop as message queue between the single JS thread and the OS.

2.2.5 Angular

Angular is a component-based JS framework that can reuse UI components and can bind data/state of objects and variables in JS class and render them in real-time. It

supports managing JSON data models for message passing between server and front-end and prevents logical and data-related errors that occur by passing incorrect data models. Angular keeps track of what's already rendered in the UI and repaints only the changes, instead of re-rendering the whole UI.

2.2.6 Electron.js

Electron.js is a tool that converts Node.js web applications into desktop applications. It combines Node.js and Chromium (Core of Google Chrome). Electron overcomes some of the limitations of Web technologies, which is why Electron is chosen for this project and the features that are not available on Web-browser and are used on this project are:

- Accessing local file System.
- Accessing Operating System modules, commands, and devices like hard-drive data, etc.
- Opening local files directly in the default system application. (eg: opening .txt file in notepad, etc.)
- Uploading files to SSH (Cut/Copy-Paste (local to SSH)) without losing directory structure.
- Downloading (Cut/Copy-Paste (SSH to local)) multiple files and folders, and also without losing directory structure.

CHAPTER 3

SYSTEM MODELING

3.1 Features

These are some of the features in the software:

3.1.1 Add Connection

User can add an SSH connection by entering the details, which will be stored permanently for connecting to Server. The user can enter the Server IP, user name of the SSH, password, and a name and description.

3.1.2 Test Connection

Checks if the connection is working by trying to connect to the server by the details added by the user. This is implemented by trying to connect to the host specified by the user with the user name and password provided by the user. If it connects, that means the test is success and if the connection fails, then the test status will also be failed.

3.1.3 Edit Connection

Modify connection details like name of connection, description, Server IP, user name, password. These details can be modified anytime.

3.1.4 Delete Connection

Removes Connection details that have been saved.

3.1.5 View Directory Contents

User can view both SSH and local files and folders and can see if the folders are filled (have content) or not, or is readable or not. These changes can be seen visually in the GUI.

3.1.6 Create New File or Folder

User can create new file or folder by right clicking and clicking on create new file / create new folder option.

3.1.7 Rename

User can rename file or folder. User can only choose a name that does not already exist in the same folder.

3.1.8 Delete

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

3.1.9 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. Available types are:

- Local to Local (Cut/Copy)
- Local to SSH (Upload)
- SSH to SSH (Cut/Copy)
- SSH to Local (Download)
- SSH to another SSH

3.1.10 Shortcuts

Keyboard Shortcuts as well as other GUI methods can be used to use the above features. eg: Ctrl+A selects all files and folders, etc

3.1.11 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.

3.1.12 Search

Search items in a directory, Partial Search is also possible. The search key match can be anywhere within the file/folder name.

3.1.13 Open File

Files in local System can be opened in default application, like text file can be opened in text editor, image can be opened in image viewer etc.

3.1.14 Open Shortcut File

In local System User can open a shortcut and if it is a file, the file in path gets opened, if it is a folder, it will open that directory.

3.2 System Architecture

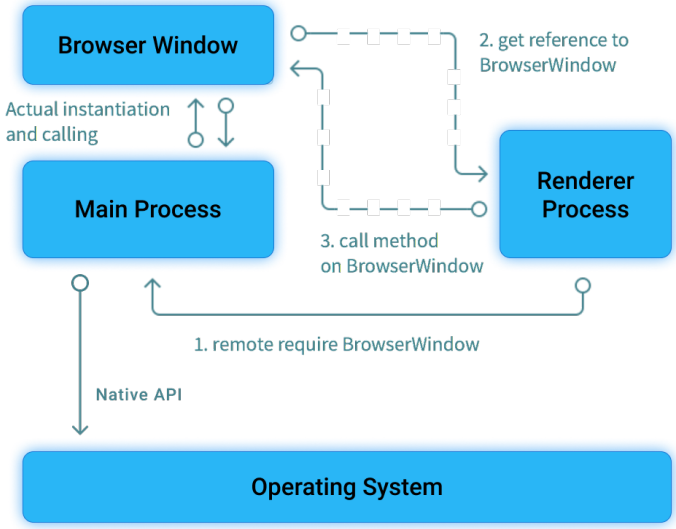

Fig. 3.1. System Architecture

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.

Along with the ipcRenderer and ipcMain of Electron used for communication, traditional HTTP requests are used using Express.js server and also WebSockets which allows realtime data passing without closing the connection.

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.

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 failure - response model

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

4.2.4 Operation partially succeeded - response model

```
{
  "process_id" : number,
  "status" : "partial-completion",
  "source": {
    "server": String, // "user@host_ip"
    "baseFolder": String // path to base folder
  },
  "target": {
    "server": String, // "user@host_ip"
    "baseFolder": String // path to base folder
  },
  "success-files": String[], // filenames of succeeded items
  "failed-files": String[] // filenames of failed items
}
```

4.2.5 Operation success - response model

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

4.2.6 Progress - response model

```
{
  "process_id" : number,
  "status" : "in-progress",
  "max": 100,
  value: number, // value of completed unit relative to max value.
}
```

4.3 Scrum Details

4.3.1 Product Backlog

Sl No.	User Stories	Priority	Estimated Time
1	User can browse local files	2	4 days
2	User can create new file	2	2 days
3	User can create new folder	2	1 day
4	User can delete files and folders	2	7 days
5	User can rename files and folders	2	5 days
6	User can open files, folders and shortcut files	2	4 days
7	User can add, edit, delete, test ssh connections	1	13 days
8	User can browse SSH files	1	2 days
9	User can create new file/folder	1	1 day
10	User can delete/rename files/folders	1	5 days
11	User can open folders	1	3 days
12	User can copy/paste files/folders	1	7 days
13	User can cut/paste files/folders	1	1 day

Fig. 4.1. Product backlog

4.3.2 Scrum Backlog

Sl No.	Sprint	To Do	Status
1	Sprint 1	Analyze Features of Existing System	Completed
2		Develop UI Wireframe and Prototype	Completed
3		UI basic backbone development	Completed
4		Create Data Models	Completed
5	Sprint 2	SSH Connection establishing	Completed
6		SSH Connection testing	Completed
7		local file System accessing	Completed
8	Sprint 3	create files feature	Completed
9		create folders feature	Completed
10		delete files and folders feature	Completed
11		browse directoy contents feature	Completed
12	Sprint 4	GUI file management feature	Completed
13		Serach feature	Completed
14		Open files feature	Completed
15	Sprint 5	Rename feature	Completed
16		Packaging for Multiple Operating Systems	Completed
17	Sprint 6	Testing in multiple Operating Systems	Completed
18		File Copy-Paste Feature	Completed
19		File Cut-Paste Feature	Completed
20		Got To directory path feature	Completed

Fig. 4.2. Sprint backlog

4.4 UI Designing

No templates or libraries are used to design UI, Used built-in Grid and Flex System of CSS. Angular is only used for data transfer between UI and back-end, and to generate dynamic elements and maintain state of components. Icons are from open-source project feather icons and illustrations and other vector graphics are drawn using figma, a prototyping tool.

4.4.1 Screenshots

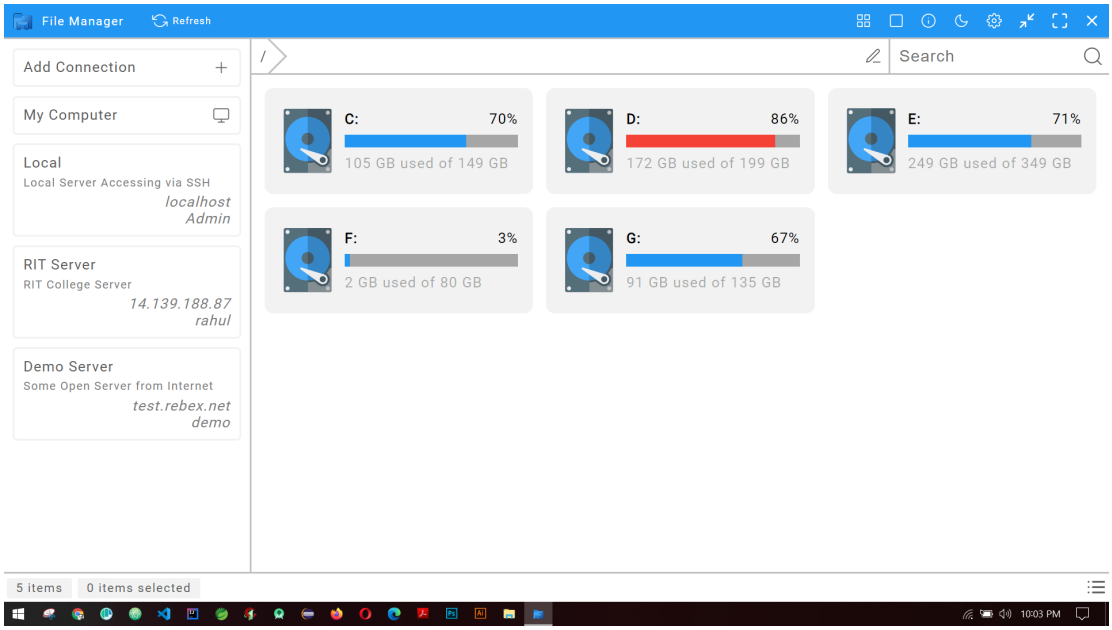

Fig. 4.3. Local Drives

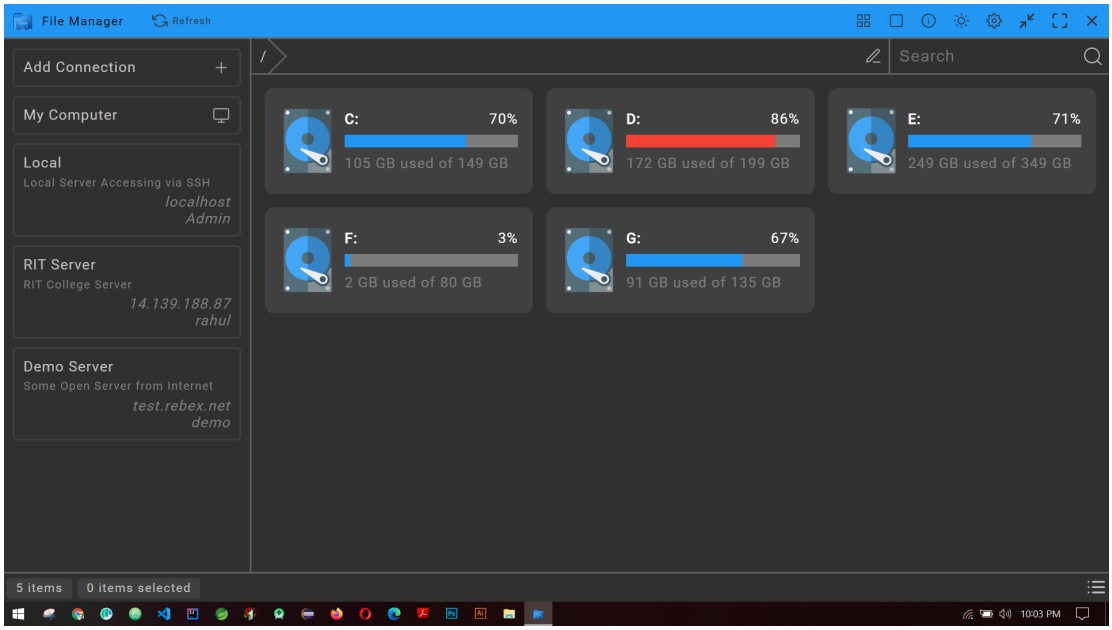

Fig. 4.4. Dark Mode

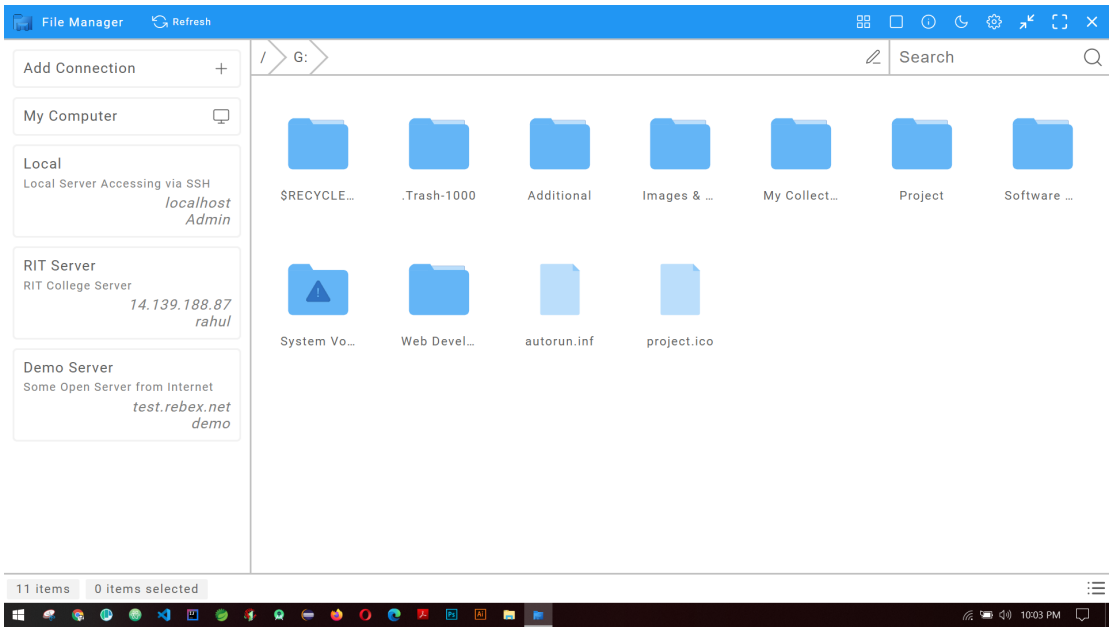

Fig. 4.5. List Directory Contents

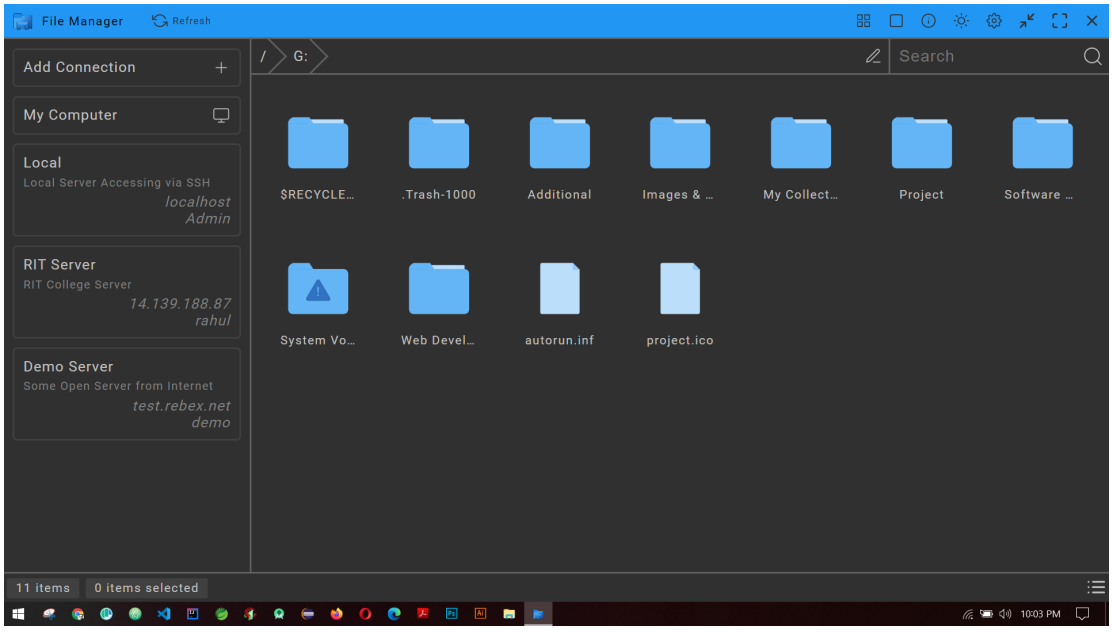

Fig. 4.6. List Directory Contents Dark Mode

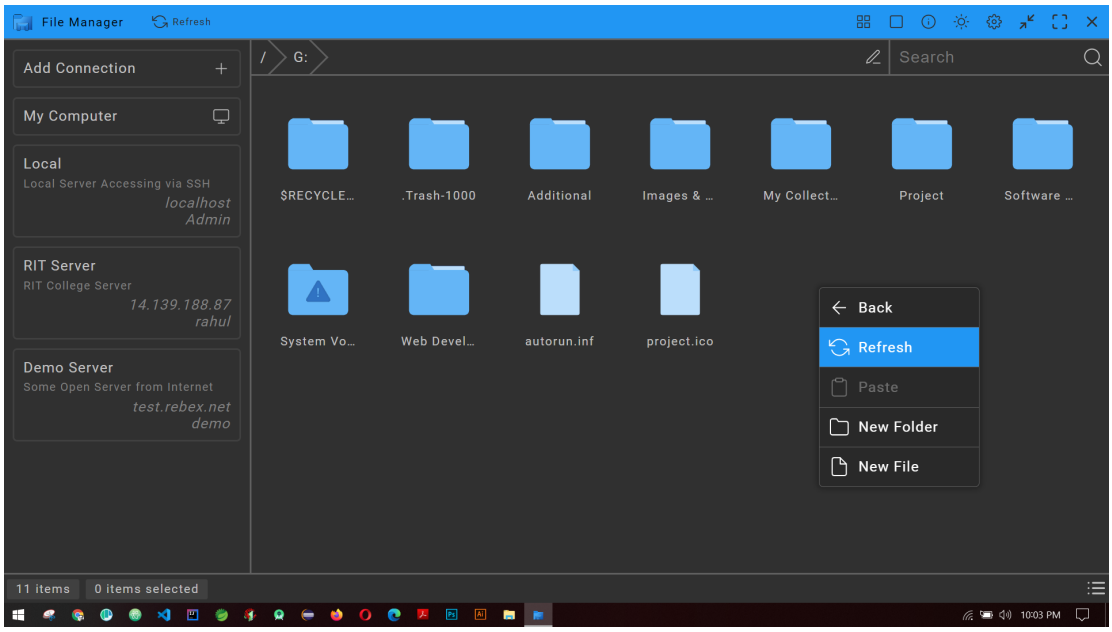

Fig. 4.7. Right-Click Options

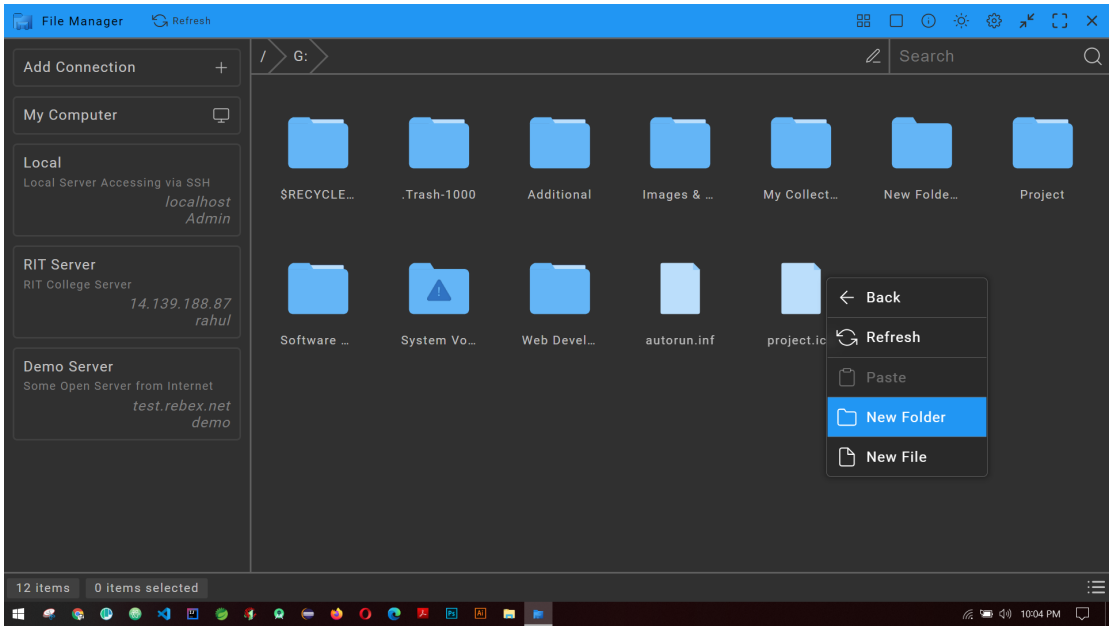

Fig. 4.8. Create New Folder

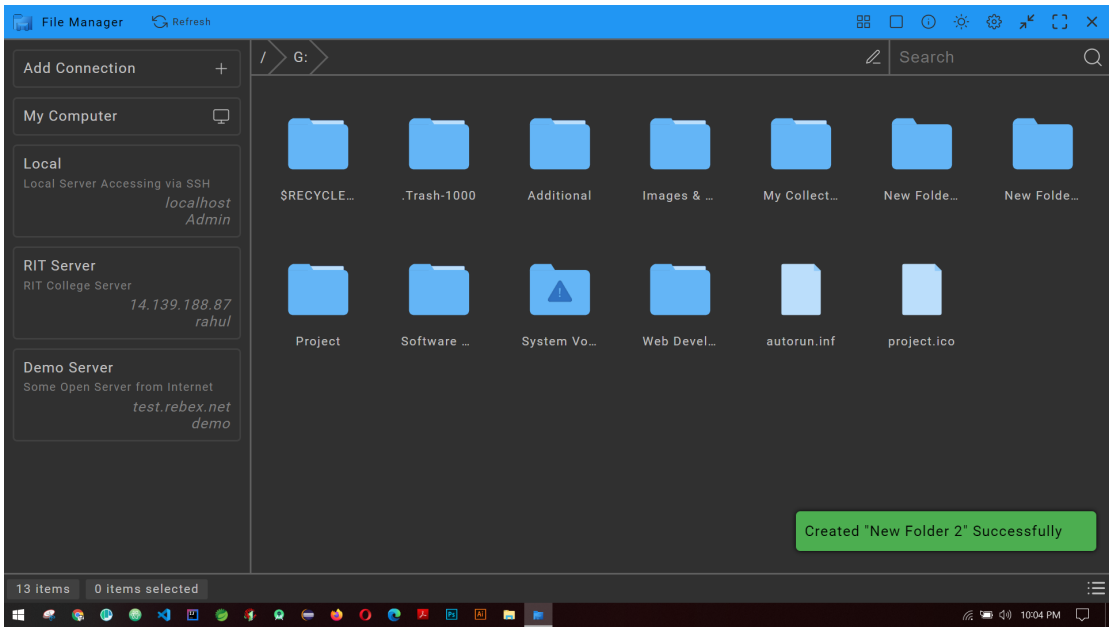

Fig. 4.9. Success Notification

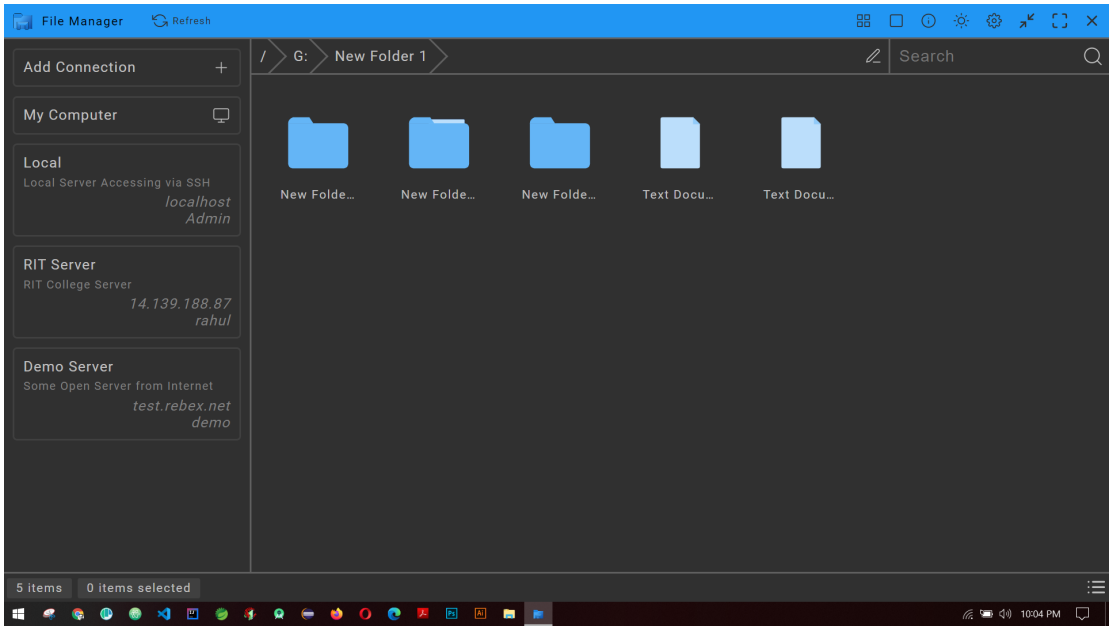

Fig. 4.10. New Folder Created

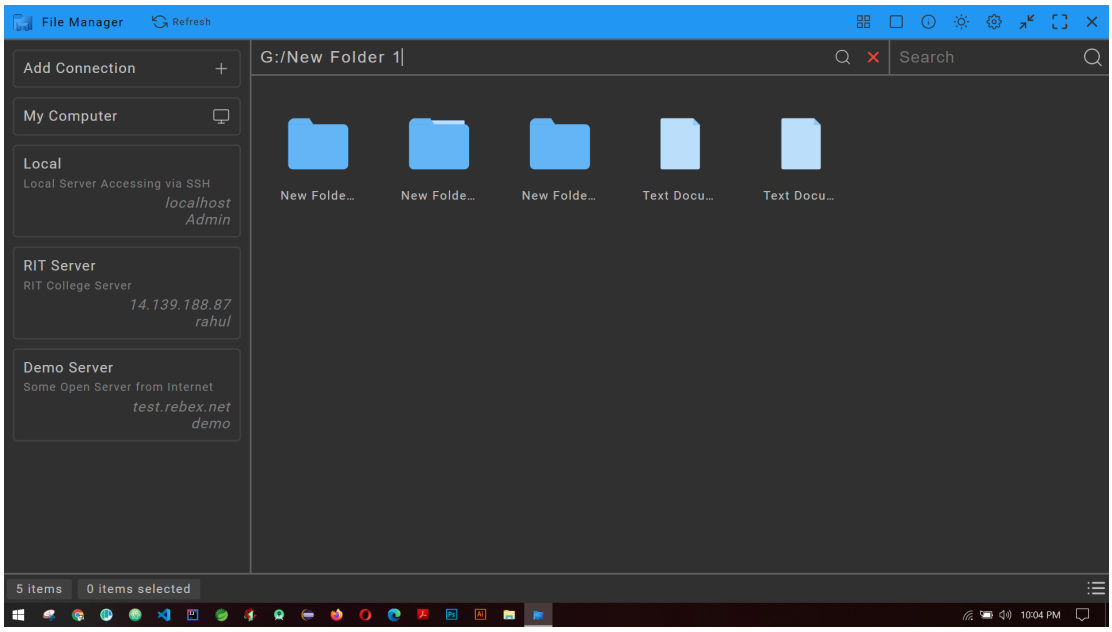

Fig. 4.11. Edit Path

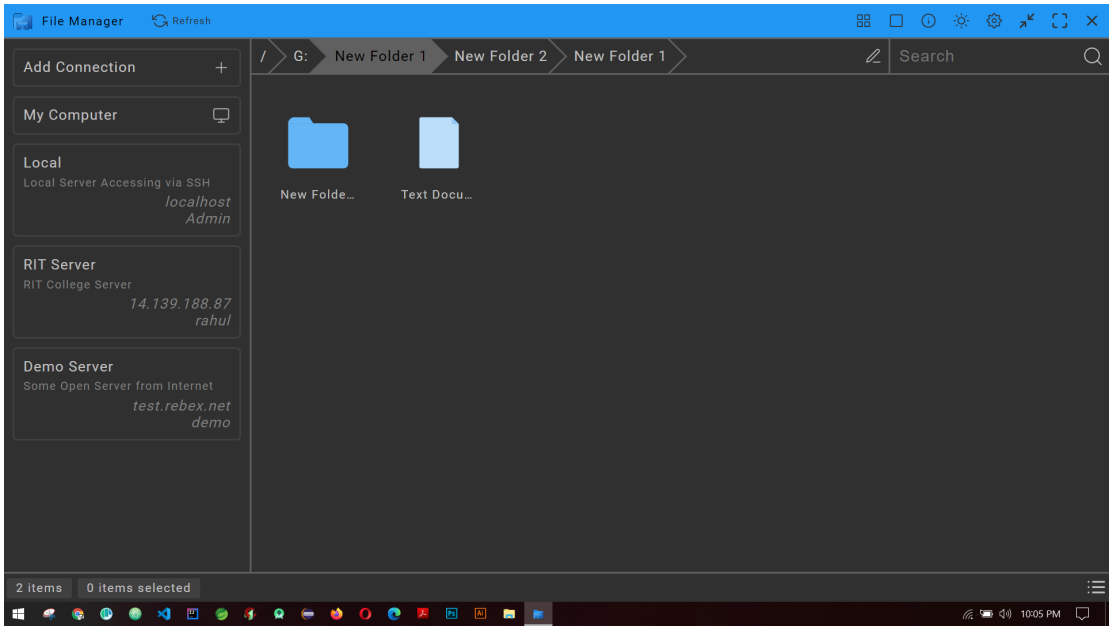

Fig. 4.12. Move to Path

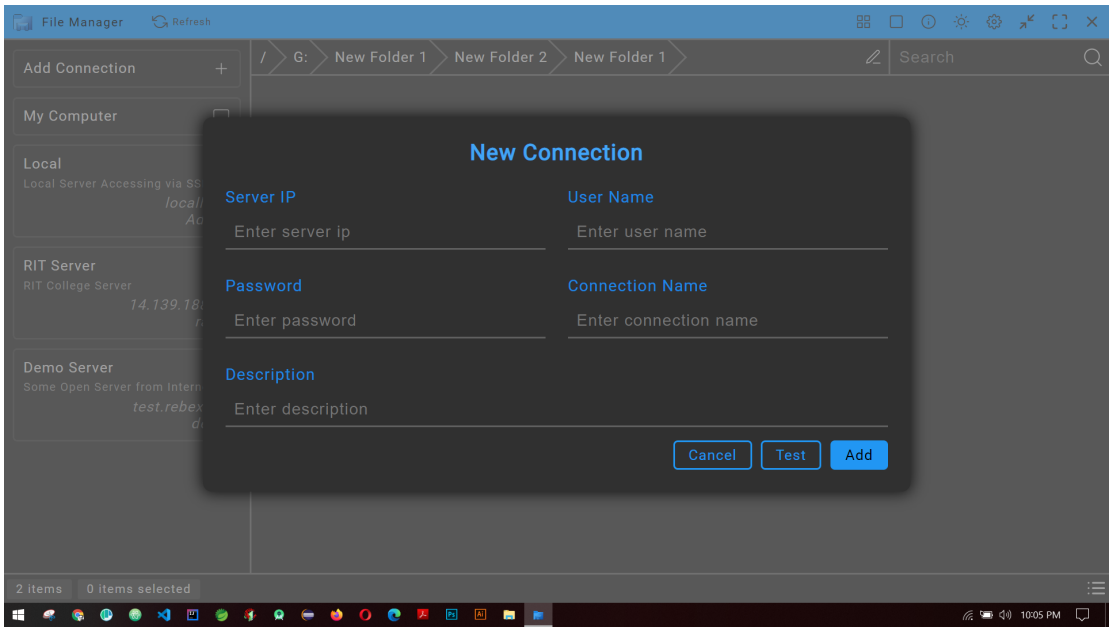

Fig. 4.13. Add Connection

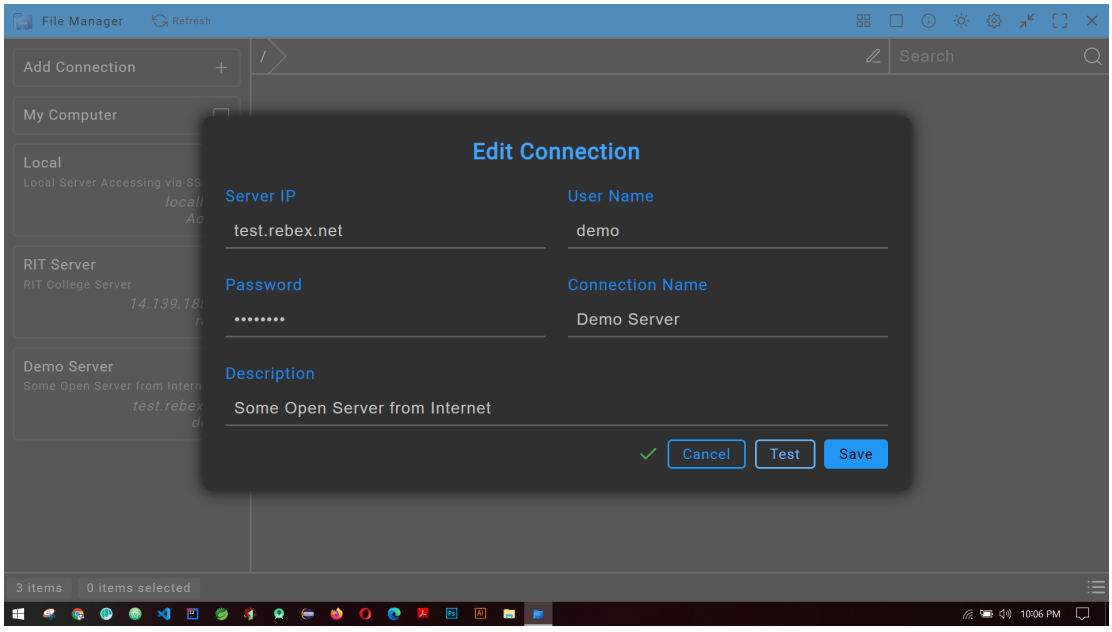

Fig. 4.14. Edit Connection

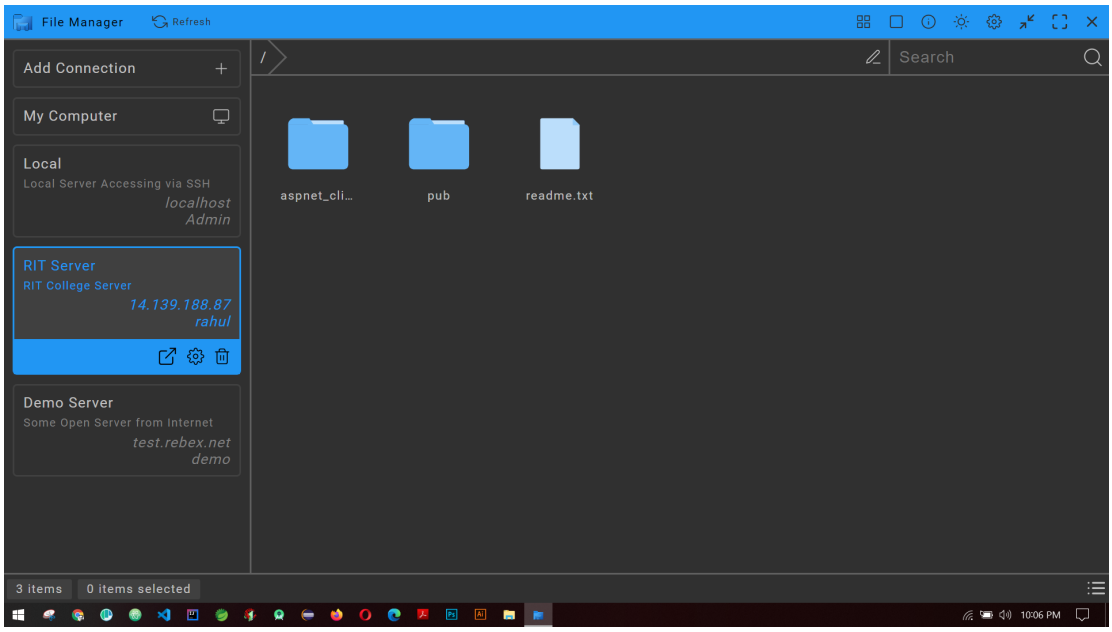

Fig. 4.15. Connect to SSH

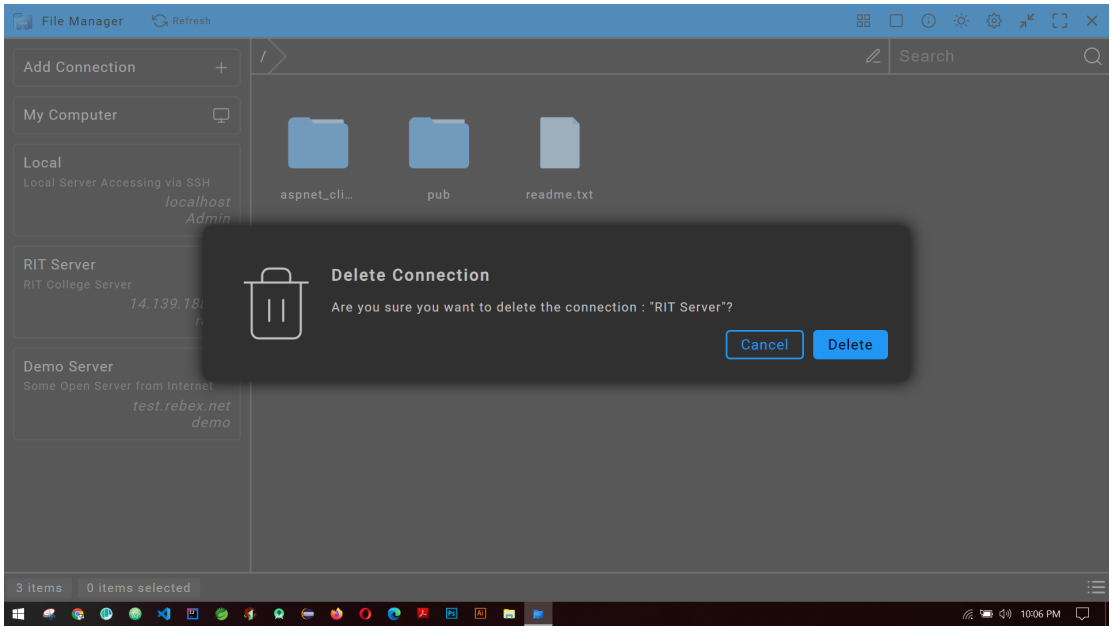

Fig. 4.16. Delete Connection

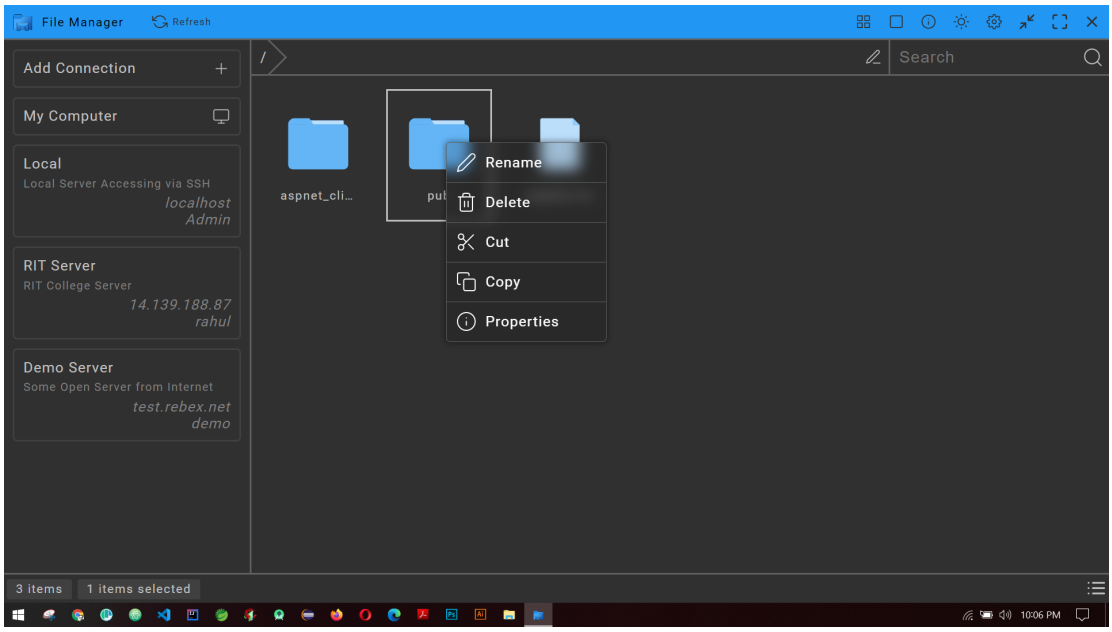

Fig. 4.17. Folder Right Click

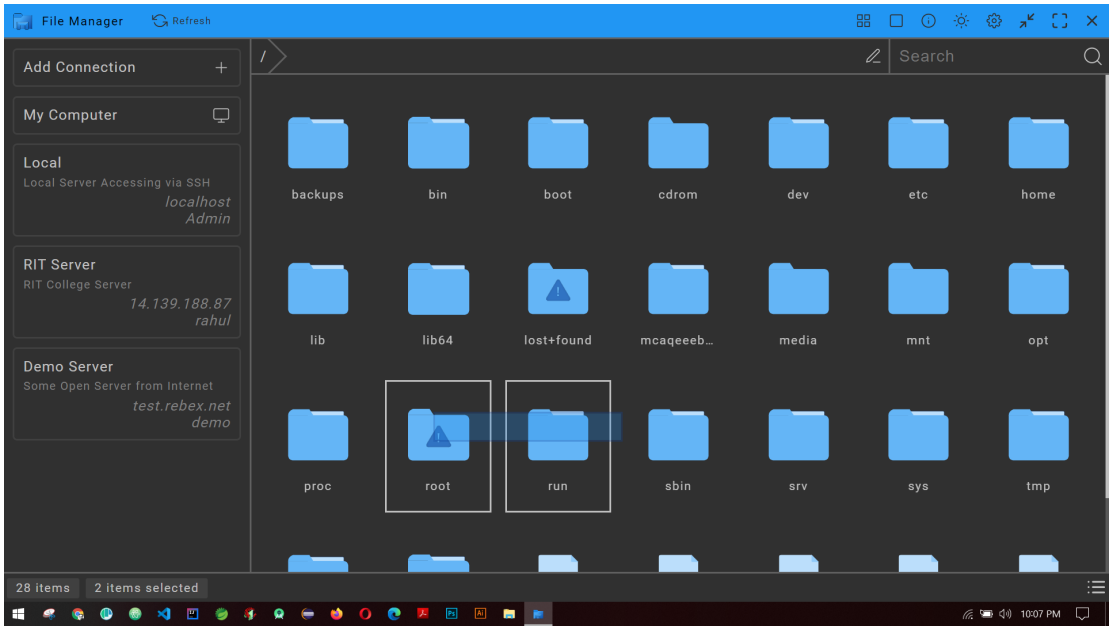

Fig. 4.18. Drag Select Files and Folders 1

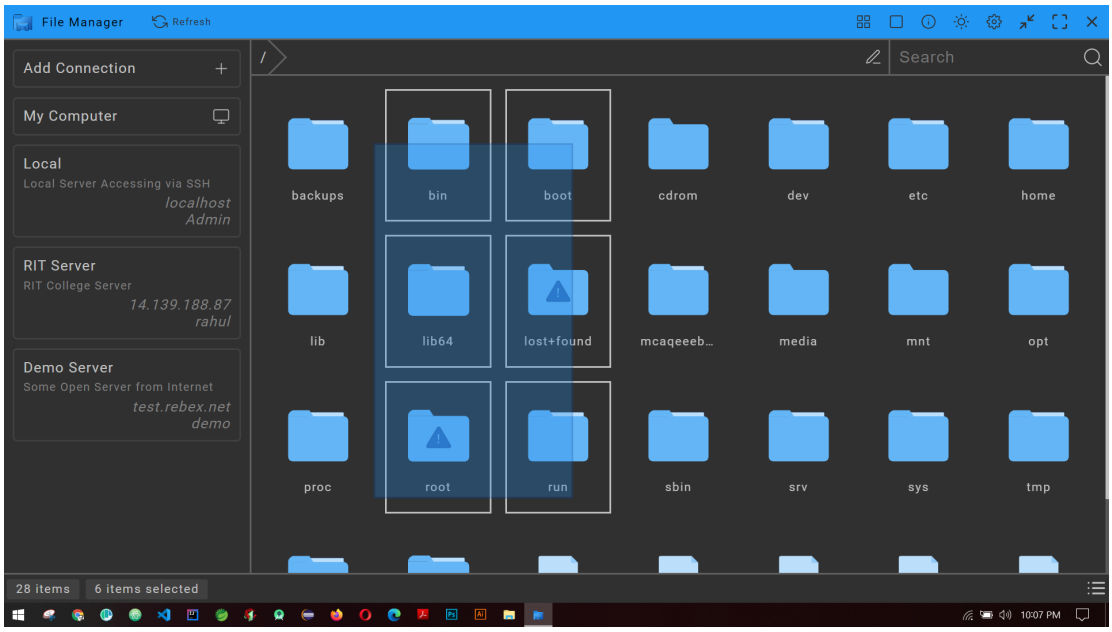

Fig. 4.19. Drag Select Files and Folders 2

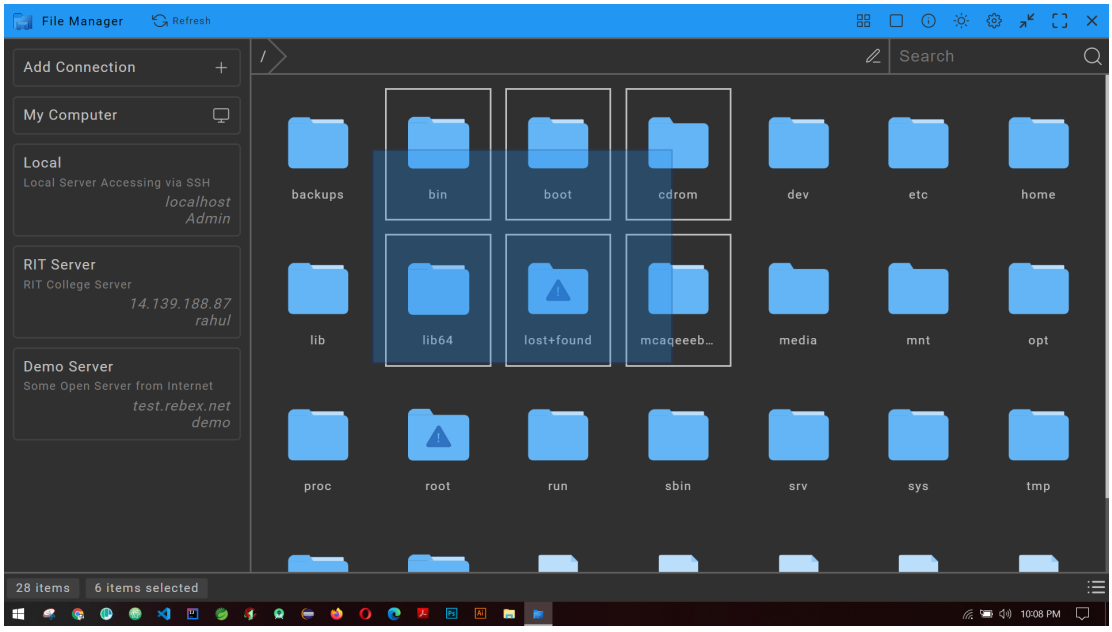

Fig. 4.20. Drag Select Files and Folders 3

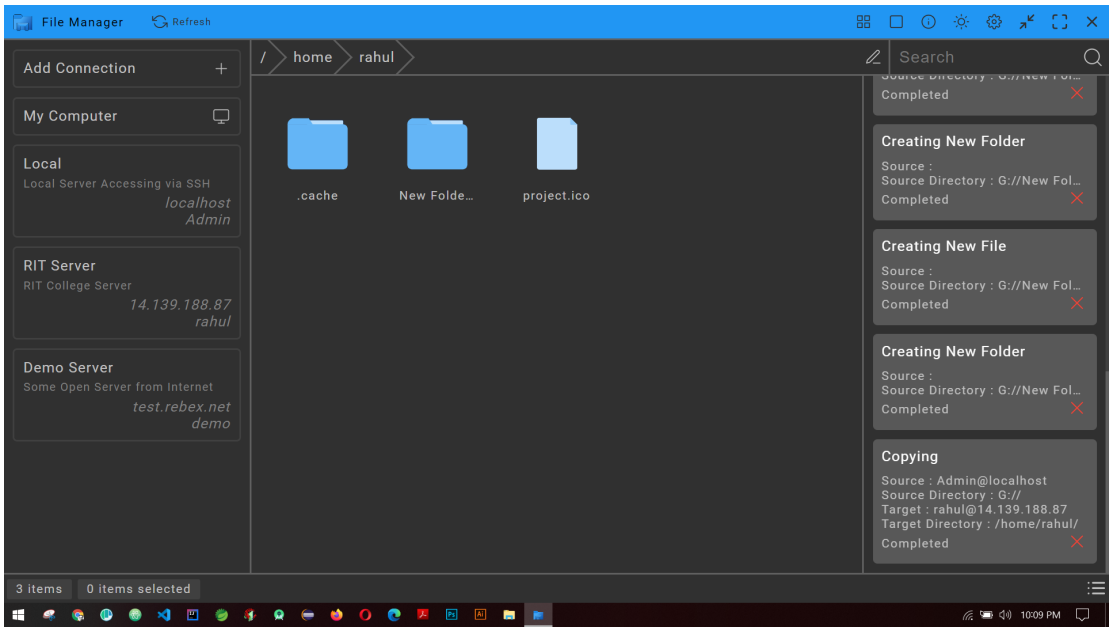

Fig. 4.21. Background Task Queue

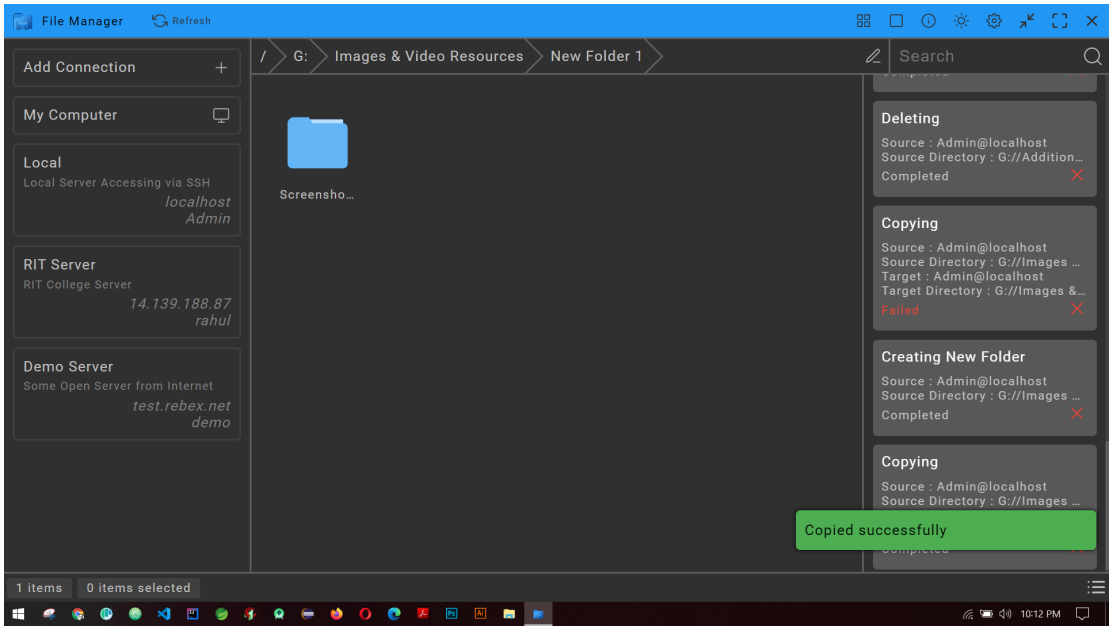

Fig. 4.22. View Old Operations

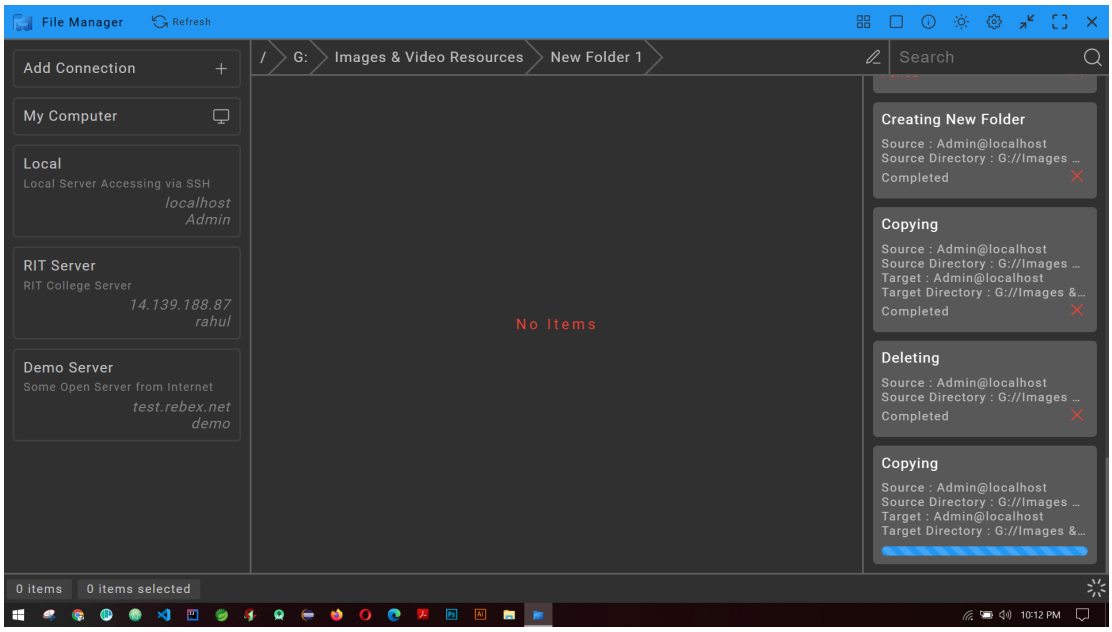

Fig. 4.23. Background Task in Progress

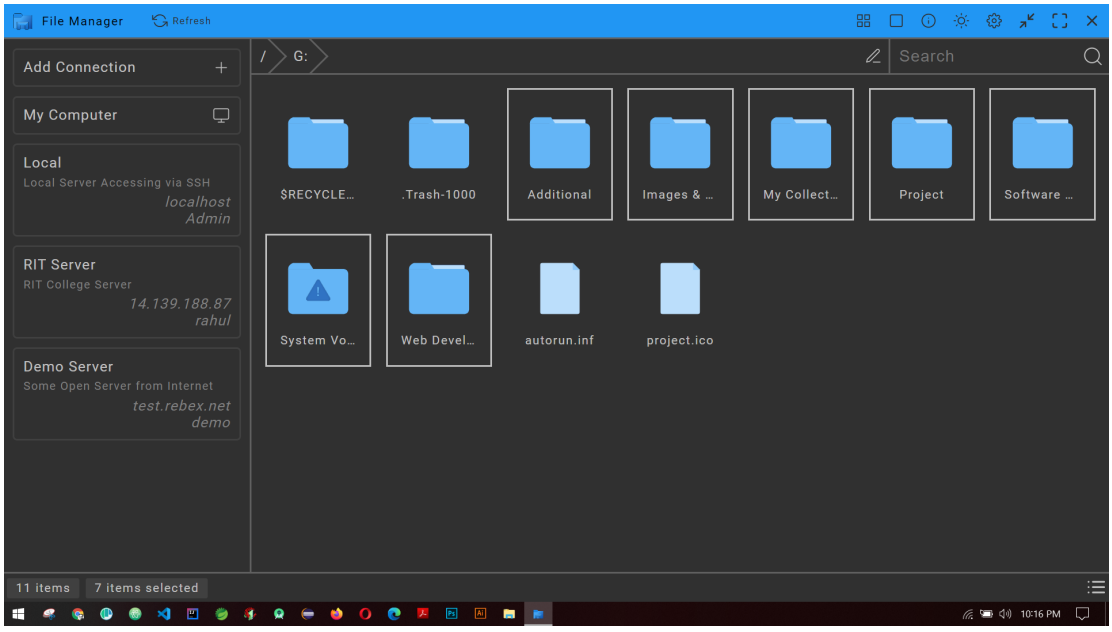

Fig. 4.24. Shift + Select

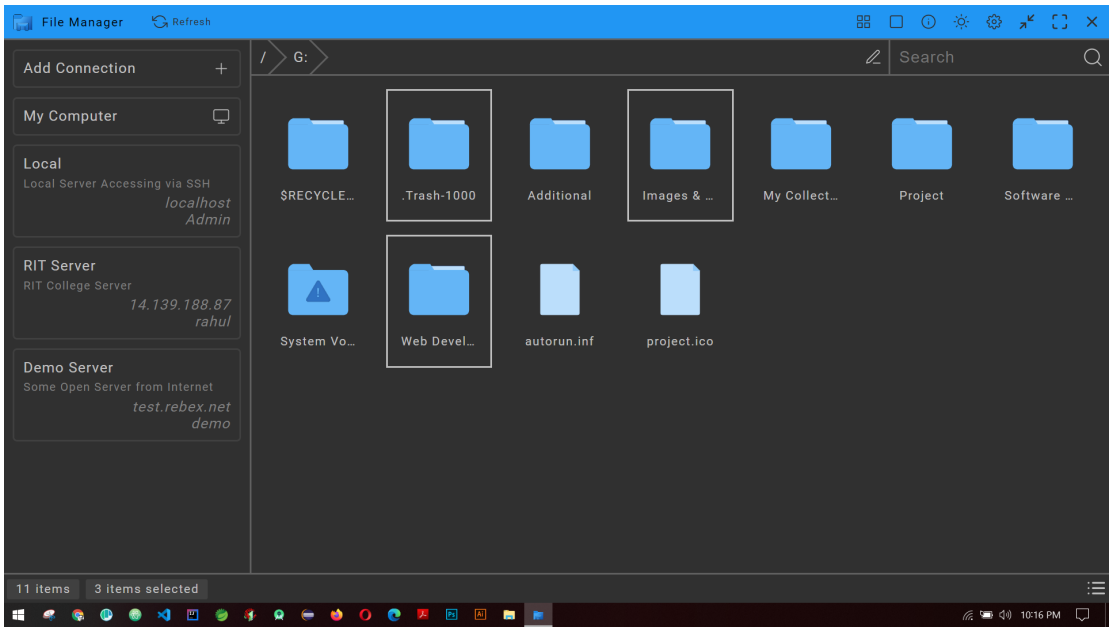

Fig. 4.25. Ctrl + Select

CHAPTER 5

SYSTEM TESTING

- Realtime error prevention methods are used by implementing data type system using TypeScript [10]. Which means that data types are enabled in JS by using TS even though Js is dynamically typed, this helps preventing 30% of the logical errors which are caused by data type mismatch.

- Other plugins used for testing are ES Lint [11] and TS Lint [12], Which shows real-time debugging messages in IDE.

- Some important functions are tested as part of unit Testing, by running against pre-defined input and their expected outputs. eg: some example file paths are set as input and expected normalized path like these:

  Input:

  C://Folder1/test/../.

  /home/folder/test_folder/../

  Expected output:

  C://Folder1/

  /home/folder/

  Some of these data are predefined and tested whenever function is altered

- Built-in assert modules were also used for testing.

- SSH connections are tested for Windows and Linux Servers.
- Manually Tested all features in Windows x64 bit OS and Ubuntu x64 bit OS.

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, temp files etc. The Software is portable, customizable and can be generated executable packages for the following types:

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

These packages are built using the open-source npm package known as electron-packager. Electron Packager is a command-line tool and Node.js library that bundles Electron-based application source code with a renamed Electron executable and supporting files into folders ready for distribution. Electron packager can generate the executable version and can also hide source-code by combining it to asar file.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

The web application is packed into executable files and 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. In future it is expected to make sure that the web version of the app also supports all the features in the application like opening a native application by opening a file, maintaining directory structure in upload/copy option or downloading multiple files and folders in browser. These operations are only available in Desktop app version, not in browser version.

REFERENCES

[1] Node.js Documentation, “https://nodejs.org/en/docs/,” Mar 2021.

[2] JavaScript Mozilla Developer Documentation, “https://developer.mozilla.org/en-us/docs/web/javascript,” Mar 2021.

[3] Express.js Documentation, “https://expressjs.com/,” Apr 2021.

[4] Angular, “https://angular.io/,” Apr 2021.

[5] Electron.js Documentation, “https://www.electronjs.org/docs,” Apr 2021.

[6] SSH2-Promise npm Documentation, “https://www.npmjs.com/package/ssh2-promise,” May 2021.

[7] SSH2 Documentation, “https://www.npmjs.com/package/ssh2,” May 2021.

[8] Atom IDE Documentation, “https://atom.io/docs,” Mar 2021.

[9] WebSocket Documentation, “https://www.npmjs.com/package/ws,” Apr 2021.

[10] TypeScript Documentation, “https://www.typescriptlang.org/docs/,” May 2021.

[11] ES Lint Documentation, “https://eslint.org/,” May 2021.

[12] TS Lint Documentation, “https://palantir.github.io/tslint/,” May 2021.