الجمهورية الجزائرية الديمقراطية الشعبية

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

وزارة التعليم العالي و البحث العلمي

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

جامعة عمّار ثليجي بالأغواط

UNIVERSITY OF AMAR TELIDJI LAGHOUAT

كلية العلوم

FACULTY OF SCIENCES

قسم الإعلام الآلي

DEPARTMENT OF COMPUTER SCIENCES

Master Thesis

Domain Mathematics and Computer Science

Field Computer Science

Option Distributed Networks, Systems, and Applications

Submitted by:
Abdelkader Babaghayou

Topic

Development of an open source tool for learning Linux

Defended Publicly in Front of the Jury Composed of :

M_s.	Leila Benarous	MCB.	PRESIDENT	UATL
M_r.	Lakhdar Kechna	MAA	REVIEWER	UATL
M_r.	Chaib NourEddine	MCA	REVIEWER	UATL
M_s.	Amel Belabbaci	MCB.	SUPERVISOR	UATL

Academic Year 2023/2024

Dedications

I would like to dedicate this humble work for my beloved parents whose constant support and belief in me have been my anchor throughout this journey. Your encouragement and patience have been invaluable, and I'm so grateful for everything you have done for me . To my family and friends, who have been with me through all the ups and downs . Thank you for being by my side and making this possible.

Abdelkader Babaghayou .

Acknowledgements

First and foremost , i would like to acknowledge and offer my recognition to Allah Almighty for all the blessings in my life .

I extend my gratitude towards my supervisor Dr. Amel Belabbaci, for her support and guidance throughout this project , im very thankful for her help and making this project possible .

I would to offer my sincere gratitude go to the committee members : Dr. Leila Benarous , Dr. Lakhdar Kechna , and Prof. Chaib NourEddine , for accepting to review my humble work .

Abdelkader Babaghayou , September 2024

Abstract

Learning Linux can be challenging for new users due to its complexity.This thesis propose an open source tool that comes as a web application designed to make learning Linux easier and more engaging especially for beginners.The application features a visual interface that helps users practice key Linux concepts and commands through an interactive terminal emulator . this tool aims to simplify the learning process and improve how beginners grasp essential Linux skills and test various commands without harming their own system . Testing showed that users found emulators to be a valuable resource, making it easier to practice and use Linux.

Keywords: Linux, E-learning, Web App

مُلخّص

يمكن أن يشكل تعلم نظام لينيكس تحديًا للمستخدمين الجدد نظرًا لتعقيده وصعوبته . لذلك تقترح هذه الأطروحة أداة مفتوحة المصدر تأتي على شكل تطبيق ويب مصمم لجعل تعلم لينيكس أسهل وأكثر قابلية للتعلم خاصة للمبتدئين. يتميز التطبيق بواجهة مرئية تساعد المستخدمين على التمرن و استخدام أوامر لينيكس الأساسية من خلال محاكي نافذة اوامر لينكس . تهدف هذه الأداة إلى تبسيط عملية التعلم وتحسين كيفية استيعاب المبتدئين لمهارات لينيكس الأساسية واختبار الأوامر المختلفة دون الإضرار بالنظام الخاص بهم.

Keywords: Linux, E-learning, Web App

Résumé

L'acquisition de connaissances sur Linux peut constituer un défi pour les débutants en raison de sa complexité. Dans cette thèse, nous proposons une solution open source qui se présente sous la forme d'une application Web, dans le but de simplifier et d'accommoder l'apprentissage de Linux, en particulier pour les débutants. Un émulateur interactif de terminal est inclus dans l'application, ce qui facilite la pratique des concepts et des commandes fondamentales de Linux. Cet outil a pour but principal de simplifier l'apprentissage et d'améliorer la façon dont les débutants apprennent les bases de Linux et testent diverses commandes sans compromettre leur propre système.

Mots-clés: Linux , e-learning , Web App .

Table of Contents

List of Figures

Abbreviations

GPL	General Public License
IoT	Internet of Things
SUDO	SuperUser Do
UML	Unified Modeling Language
MVC	Model View Controller
GC	Garbage Collection
API	Application Programming Interface
OS	Operating System
RPC	Remote Procedure Call
HTTP	Hypertext Transfer Protocol
IIS	Internet Information Services
SPA	Single-web Page Application
IDE	Integrated Development Environment
CPU	Central Processing Unit
RAM	Random Access Memory
VPS	Virtual Private Server

Chapter 1

Introduction

1 Context

Linux serves as an open-source operating system kernel that underpins a wide array of devices, ranging from personal computers to servers and embedded systems. Initiated by Linus Torvalds in 1991, Linux has evolved into a powerful and adaptable platform, bolstered by contributions from a worldwide community of developers and enthusiasts. In contrast to proprietary systems, Linux is distributed under the GNU General Public License (GPL), which permits individuals to access, modify, and share the source code. This open development approach encourages innovation and personalization, resulting in a rich ecosystem of distributions designed to meet diverse requirements[1]. Notable distributions such as Ubuntu, Fedora, and Debian provide user-friendly interfaces and robust support communities, making Linux approachable for both novices and experienced users. The operating system's stability, security, and versatility have established it as the preferred choice for numerous enterprise settings, web servers, and supercomputers. Additionally, its significance in cloud computing and the Internet of Things (IoT) highlights its critical role in contemporary technology. Although Linux may pose a learning curve for those familiar with more conventional operating systems, the abundance of online resources, forums, and documentation available helps alleviate this difficulty. The ongoing development of Linux and its dynamic community ensure its continued relevance and adaptability in a rapidly evolving technological environment[2].

There are many approaches for learning and mastering Linux for students, each suited to different learning preferences and styles. Traditional classroom courses offer a structured educational setting with direct interaction with instructors and students. Such courses can be particularly advantageous for individuals who benefit from in-person engagement and require a more guided learning experience. Conversely, self-directed study methods, including literature, online tutorials, and community forums, provide flexibility, enabling learners to advance at their own pace.Resources like "The Linux Command Line"

by William Shotts [3] and "Linux Basics for Hackers" by OccupyTheWeb[4] deliver thorough insights into Linux commands and principles, while platforms such as the Linux Foundation's training resources offer extensive knowledge and practical exercises.

In recent years, e-learning platforms have emerged as effective tools for acquiring and mastering Linux skills. Platforms such as Coursera, Udemy, LinkedIn Learning, and edX provide a variety of online courses and certifications tailored to different proficiency levels, from novices to advanced users. These e-learning platforms offer significant advantages to learners.

2 Projects objective

In the context of learning Linux , our main goal or objective concerning this project , is to make a simple tool that emulates a Linux terminal, so students or users in general can find a safe environment to practice linux commands they learned without having the risk to try it in their own system, without any sort of limitation.

3 Thesis structure

In this thesis, we discuss throughout the three next chapters every details concerning this project.

In the first chapter (Existing tools for learning Linux) ze get to see some similar tools that already exist in the internet,and the advantages these tools provide for the users, and the limitations they strict the users with.

In the second chapter (Project's Design and Analysis) we see the project structure in details, and see a demonstration of the user interaction with our tool.

In the third chapter (Implementation) we go through every detail on how we implemented the design of our project and what tools we used to work on this project.

Chapter 2

Existing tools for learning Linux

In this chapter we see an overview of a similar tools that already exist and used by students on the internet, and after that we get to see a presentation of our own tool.

1 CoCalc

CoCalc offers a full, collaborative, real-time synchronized Linux Command Line Terminal in the browser as a web application,some of the benefits that CoCalc offers to the user is having the same terminal that can be opened by multiple users at once, all users can see the same view, which adaptively resizes to a common size , You don't have to install and maintain any software to use their Linux terminal , users can also edit and work on shell script files ... ect[5].

The following figure 2.1 shows CoCalc linux terminal emulator where we try and execute two simple command "echo" and "date" :

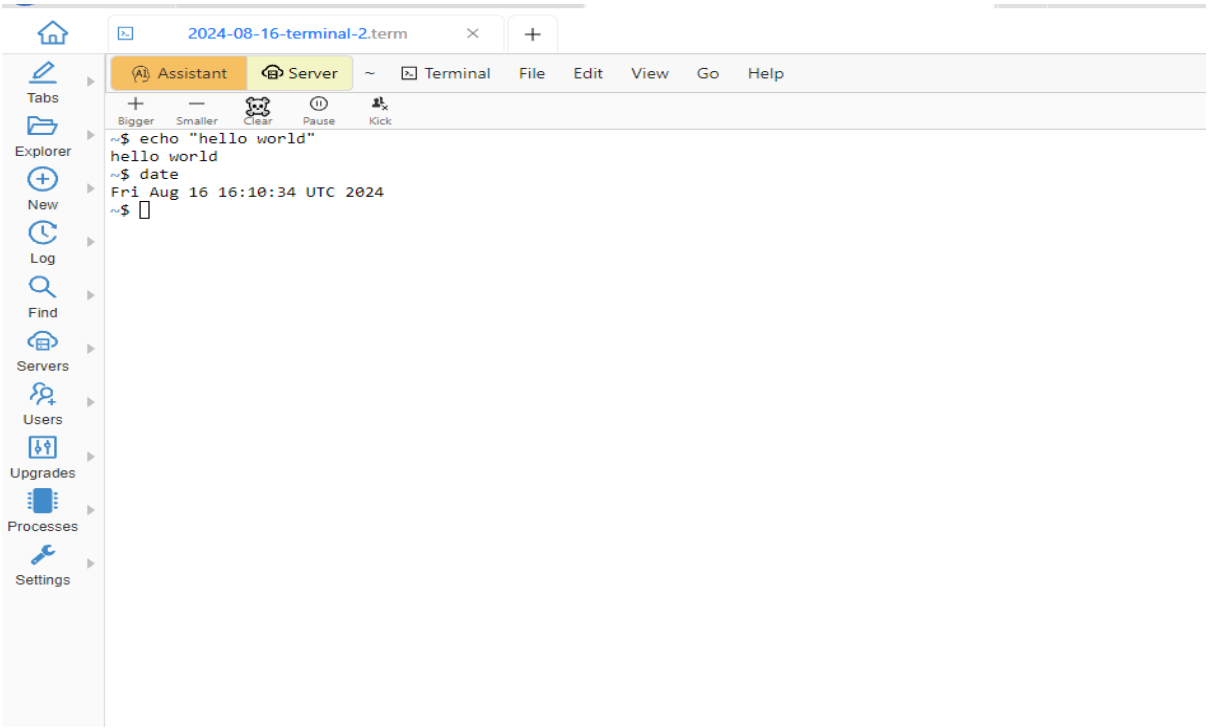

Figure 2.1: CoCalc's Linux Terminal.

We can see that in the CoCalc linux emulator user interface that we have the freedom of choosing number of processors, and creating multiple users, and working on multiple tabs, along with other benefits.

The following figure 2.2 shows execution of the commands "mkdir home" and "touch file.txt" in Cocalc linux terminal that creates a directoory called "home" and a text file called "file.txt" , and a successful attempt of deleting both the directory and the file :

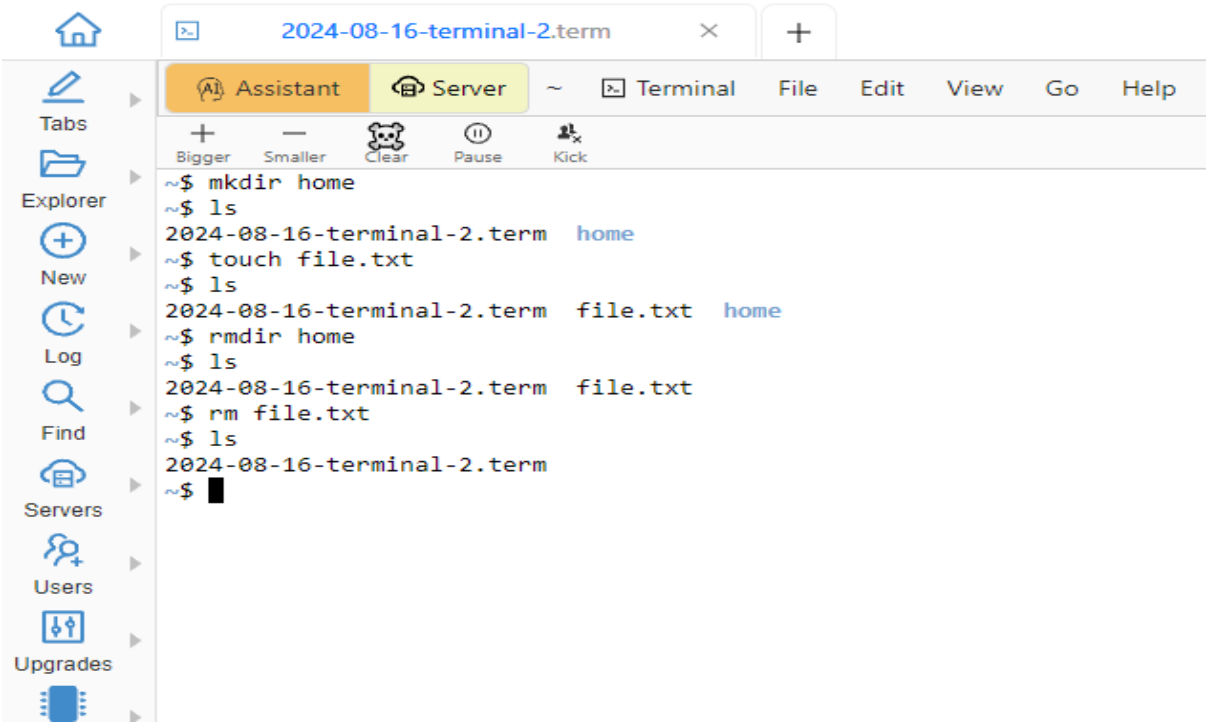

Figure 2.2: CoCalc demo.

However CoCalc Linux Terminal emulator unfortunately denies the user from using a "sudo" command, but the user can always use commands of removing directories and file manipulation.

In the figure 2.3 we have an example of using a command that require a "sudo" privilege (sudo touch file.txt) , also we can see the working directory is /home/user by executing the command "pwd" , and we can clearly see that the application forbids the user from performing a "sudo" command because the user do not have administrator privilege and due to security precautions :

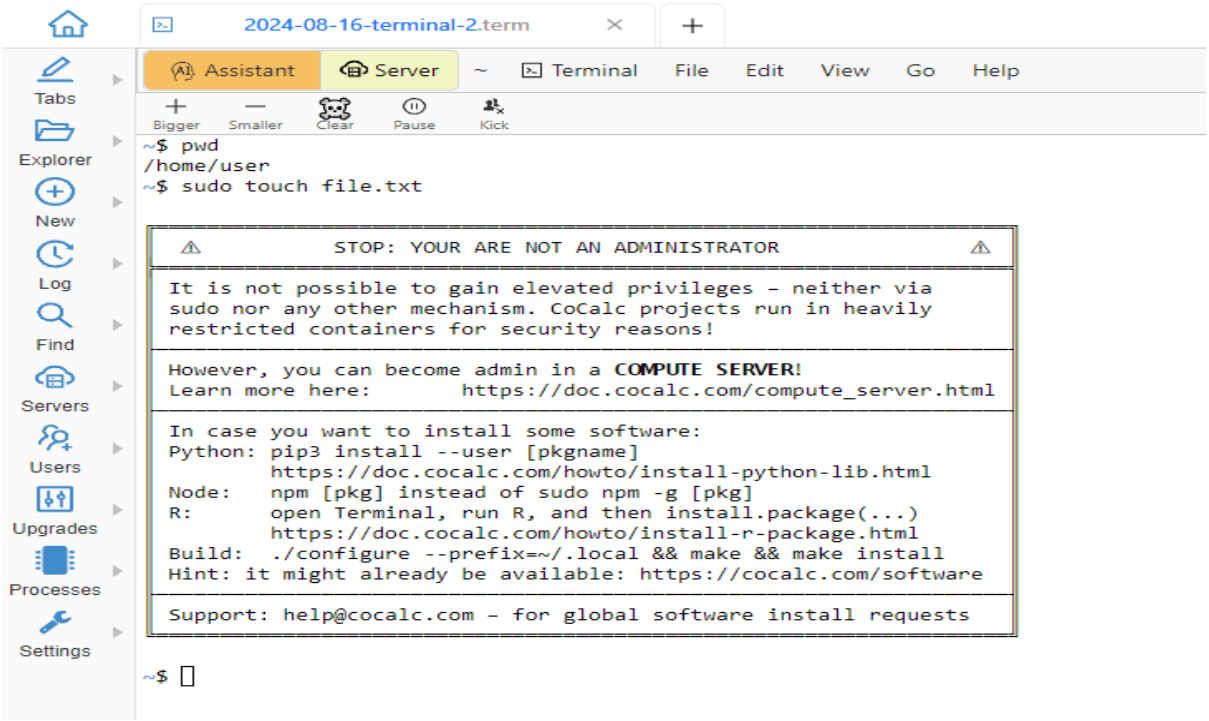

Figure 2.3: Sudo in CoCalc.

2 tutorialspoint

TutorialsPoint is an e-learning platform that offer courses for programming languages and many other features , and among the tools that TutorialsPoint offers is an online linux terminal for the beginer users to start learning the basic commands and apply what they learned the courses of TutorialsPoint [6].

The following figure 2.4 demonstrate the Linux Terminal emulator provided by TutorialsPoint :

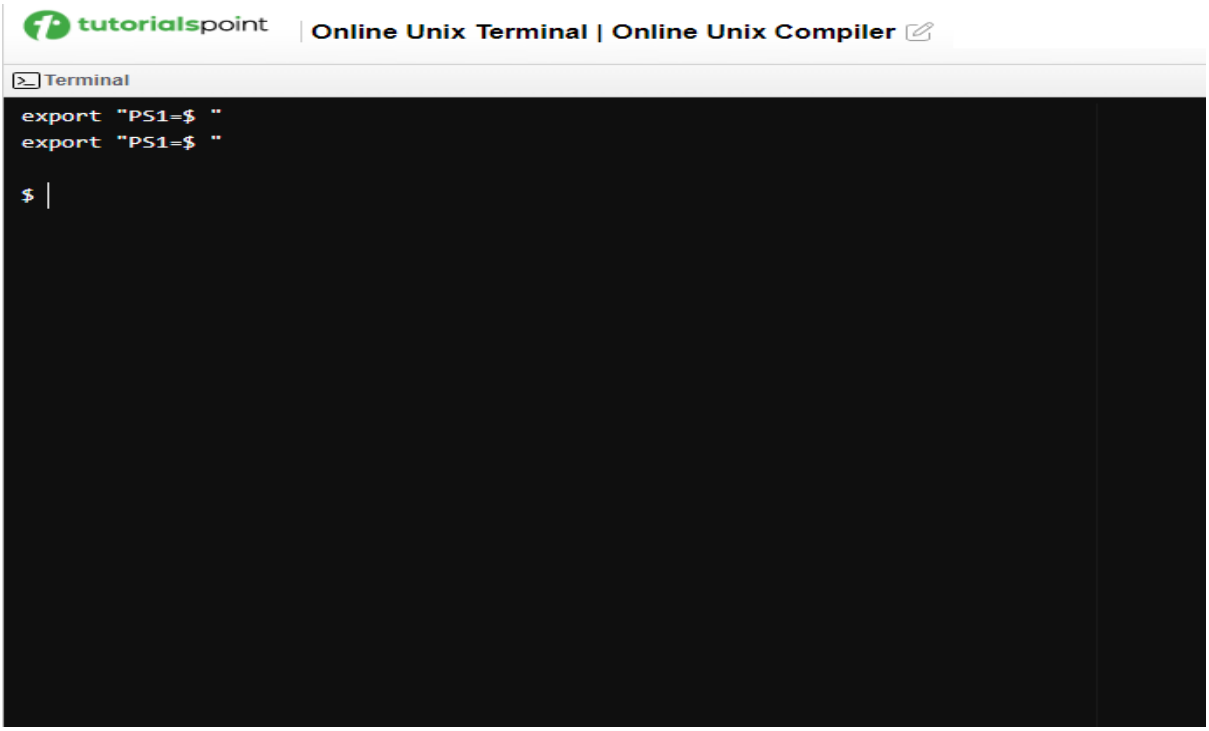

Figure 2.4: TutorialsPoint's Linux Terminal.

We can see that TutorialsPoint provide a very simple user interface of an online linux terminal that gets the job done without any extra tools.

TutorialsPoint also offer a great benefit of for their user by giving them a a root privilege .

The following figure 2.5 shows the working directory in TutorialsPoint linux terminal, after we execute the command "pwd" , and we can see the result returns /home/cg/root/ :

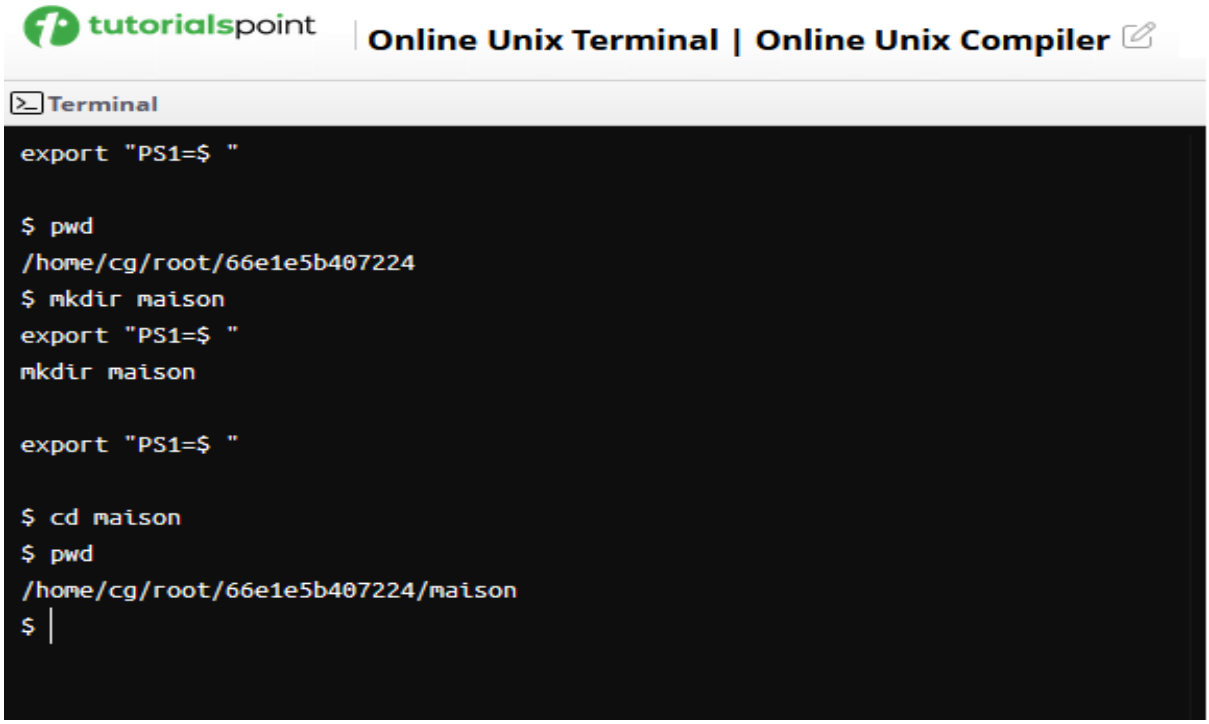

Figure 2.5: working directory.

The following figure 2.6 also shows a successful execution of simple manipulation commands , where we create a directory "home" the command "mkdir home" and creating a text file using the command "touch file.txt" , and after the creation we tried deleting both using "rmdir" and "rm" commands :

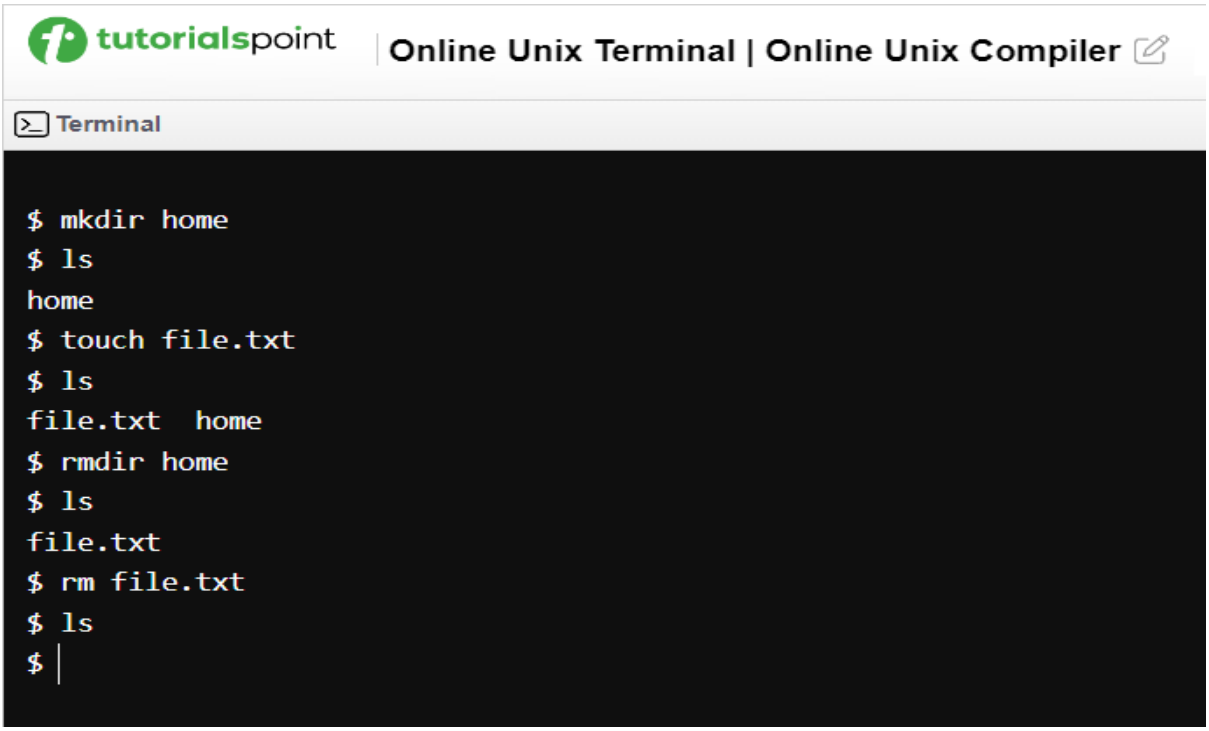

Figure 2.6: Manipulation commands.

3 Other similar Web Applications

There are many other platforms that provide a Linux Terminal emulator , on top of the list we have :

terminaltemple : simple web application that represents a simple linux terminal emulator which offer similar privileges as the other mentionned platforms , url : https://www.terminaltemple.com/

webminal : webminal is also a virtual platform that privide a space for learning linux commands among other things , url : https://www.webminal.org/

acceleratron : acceleraton offers various tools and service among those tools a linux terminal emulator also with similar advanteges as other platform , url : https://acceleratron.in/vm-linux-terminal

4 Problem Statement and Objective

In this chapter we have discussed and tried two of many other platforms that offer an online Linux Terminal emulator , and learned some of the benefits they have in common. So our main objective is to provide an open source online Linux terminal emulator that serves the same purpose with the previous tools we have seen , where users can have the same functionalities as their own system and have the full access rights and privileges to perform any command they want to explore without having the risk to corrupt their system , and have the freedom to work and upgrade on our web application.

5 Project Presentation

The following figure 2.7 shows what the user interface of our Project or our Web Application looks like :

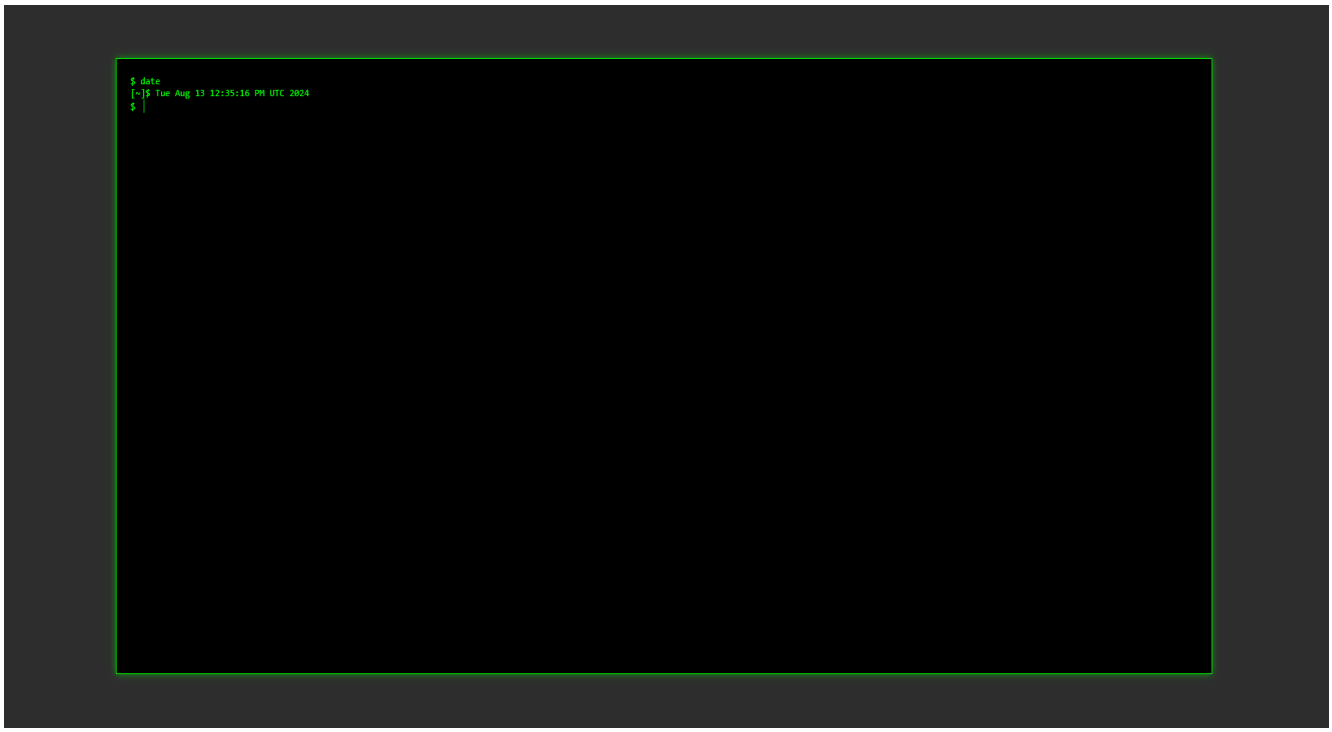

Figure 2.7: Application overview

The figure 2.8 shows a successful execution of "ls" command which displays two folders ("dossier" , "Downloads") and a file ("file.txt") , and also a successful execution of making a new folder ("home") using a SUDO privilege "sudo mkdir home" :

Figure 2.8: Application overview

6 Conclusion

After we have seen two of the most appealing similar tools to ours on the internet (Cocalc and TutorialsPoint), and seen what both of these platforms offers to their users, although we presented the primitive version of our tool, the main goal is to combine both of the advantages of CoCalc and TutorialsPoint in a single web application, it is a big project after all and require high knowledge in web development in general among other technologies to reach the final purpose.

Chapter 3

Project's Design and Analysis

This chapter looks at the project's design as represented via diagrams and design patterns that organize and separate the code.
For our project, we chose to study the application and how it works using a sequence diagram, and to structure the project's implementation before writing the actual code in a design pattern called MVC (model view controller) architecture for a cleaner and more organized code .

1 Clean Architecture

The goal of Robert C. Martin, also known as "Uncle Bob,"'s Clean Architecture architectural pattern is to maintain software projects' code simple, modular, and free from outside influences. It separates the application's business logic from external components like databases, frameworks, and user interfaces [7][8].

1.1 Why is Clean Architecture important

There are several benefits associated with Clean Architecture[7] :

High maintainability:

business logic and technology aspects are clearly separated, the code is easier to understand and update[7].

Adaptability:

It is easy to add new features or modify existing ones thanks to a well-structured design, all without affecting the overall integrity of the program[7].

Testability:

The well-organized division of components improves the software and facilitates the development of automated tests[7].

Independence from external influences:

The use of abstractions and interfaces of the program allows the program to remain flexible and independent of any technology or framework[7].

1.2 The benefits of Clean Architecture

clean architecture has many benefits that includes[7] :

Clear Separation of Responsibilities:

A clean architecture defines the responsibilities of different components, resulting in better-organized code[7].

Reduces Dependencies:

A clean architecture minimizes dependencies between different parts of the application, thereby improving flexibility and maintainability[7].

Facilitates Testing:

The clear separation of components allows testing to focus on individual parts of the application, improving testability and quality of the software[7].

Testability:

Supports Scalability and Extensibility: The modular structure of a clean architecture makes it easy to scale the application and add new features without having to rethink the entire architecture[7].

1.3 MVC(Model View Controller)Design Pattern

The Model-View-Controller (MVC) architectural pattern separates an application into three main groups of components: the model, the view, and the controller. This pattern provides separation of concerns. Through this pattern, user requests are routed to the controller, which is responsible for working with the model to perform user actions and/or retrieve query results. The controller chooses which view to display to the user and provides them with all the model data they need[9].

The following diagram in the figure 3.1 shows the three main components and how they relate to each other :

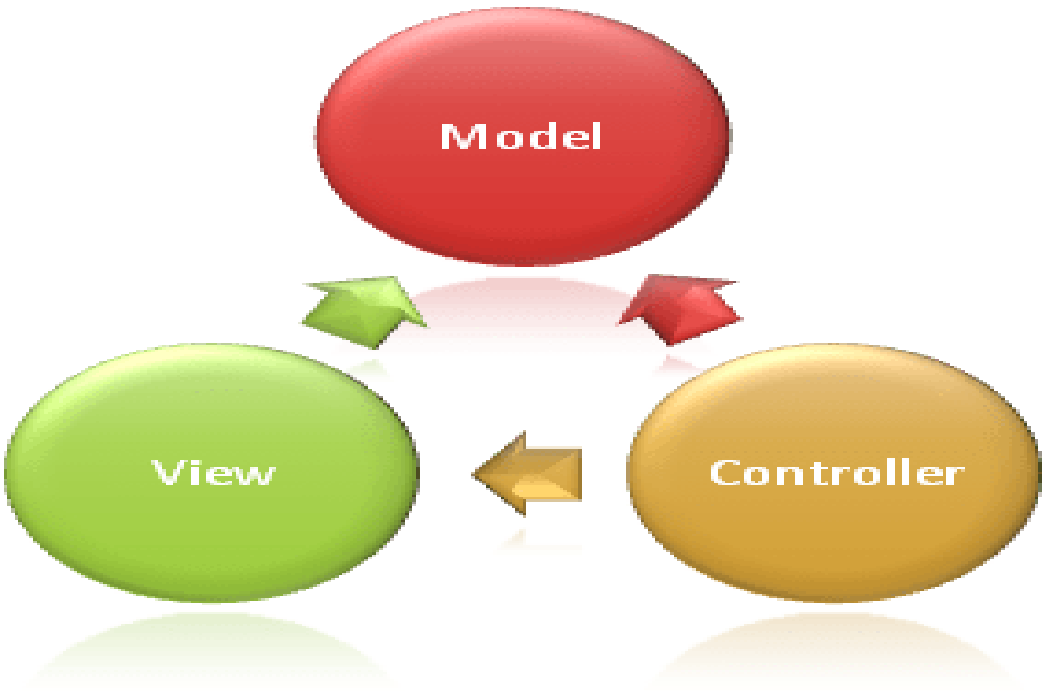

Figure 3.1: MVC structure.

Describing this responsibility helps you scale your application in complexity because it is easier to code, debug, and test something (model, view, or controller) that has a single responsibility. It is more difficult to update, test, and debug code that has dependencies spread across two or more of these three areas. For example, user interface logic tends to change more frequently than business logic. If the presentation code and business logic are combined in a single object, the object containing the business logic must be modified every time the user interface changes. This often introduces bugs and requires retesting the business logic after every minimal change to the user interface [10].

How we used MVC in the project:

In our project we used MVC as a design pattern to separate each component for a well organized code , however for the current version of our application the use of models isn't needed until future upgrades.
Concerning the front-end we separate it in the views section which sends and receives data from the controller which occupy the back-end logic .
The controller contain the code that recieves the linux commands from the view and send them to the server to execute , and retrieve the output to send it to the view once again, we can see a pseudo code in the figure 3.2 that shows how the controller works :

```
Action ExecuteCommand (string command){
    if (server.isConnected = false){
        Ssh.Connect("server@ip","passwd");
    }
    output = server.execute(command);
    return output;
}
```

Figure 3.2: Controller pseudo code

2 Sequence Diagram

UML[11] Sequence Diagrams are interaction diagrams that show the steps involved in performing an operation. They depict how items interact with one another when working together. Sequence diagrams are time-focused and visually depict the sequence of the exchange by displaying the time, what messages are transmitted, and when on the vertical axis of the figure [12].

In the figure 3.3 we can see the sequence diagram of our project and how the user interact with the application and the server :

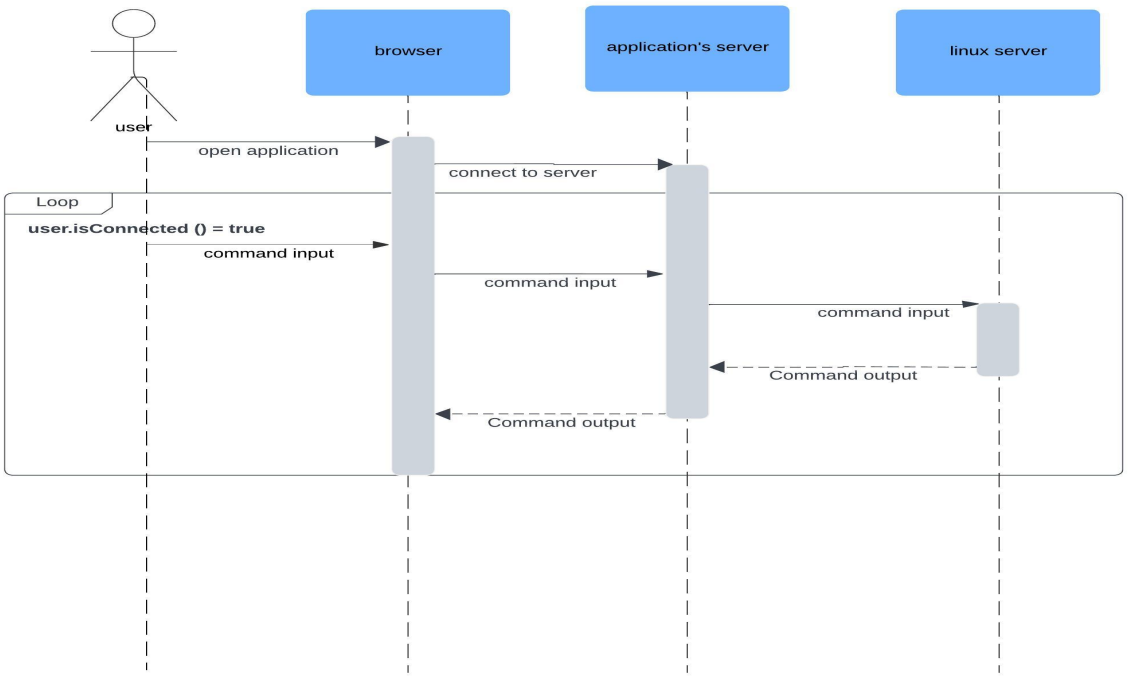

Figure 3.3: Sequence Diagram.

3 Conclusion

In this Chapter, we have discussed the functionality of our Web Application in an abstract way or in a form of a sequence diagram, which will be the basis of the next step.
next step : implementation, which we will see in the next chapter.

Chapter 4

Implementation

In this chapter of this thesis, we discuss how we implemented our application. Starting with implementing a clean architecture and what we choose as a design pattern (MVC). Next, we get see which packages and libraries we use in our application, and we also see an overview of our web application and explain how the code works.

1 Clean architecture

First of all, we all know that working alone on a project would be an easy and smooth experience. However, this is not always the case in the industry. One of the essential skills for a software engineer is the ability to work on large projects and collaborate with large teams of people.
And some of the key areas of software engineering that will make collaboration much easier are maintainability and readability of the code. Basically, if the code is written in a clean and systematic way, you will be able to add new functionality without struggle.
For web applications , generally we need to take control over three main components . which are:

• User Interface : this component represents the front-end part of our web application and in the code structure we call it the View .
• Business Logic : this component represents a part of the back-end part that is responsible for each interaction the user make with the application , in our code structure we call it a Controller .

• fetch Data : this component also represent a part of the back-end code that occupy all kind of data manipulation and interaction with the database , and we call this part

a Model , however implementing a database in our web application is saved for a future enhancements to our project , so we are not implementing Models in our code, because there is no use for them in the meantime .

1.1 Folder structure

in our application we chose as a framework Asp.net core 6 that we explain in the following sections in this chapter, this framework uses MVC as a design pattern , which separate each module or component from another in an organized manner which we can see in the figure 4.1 :

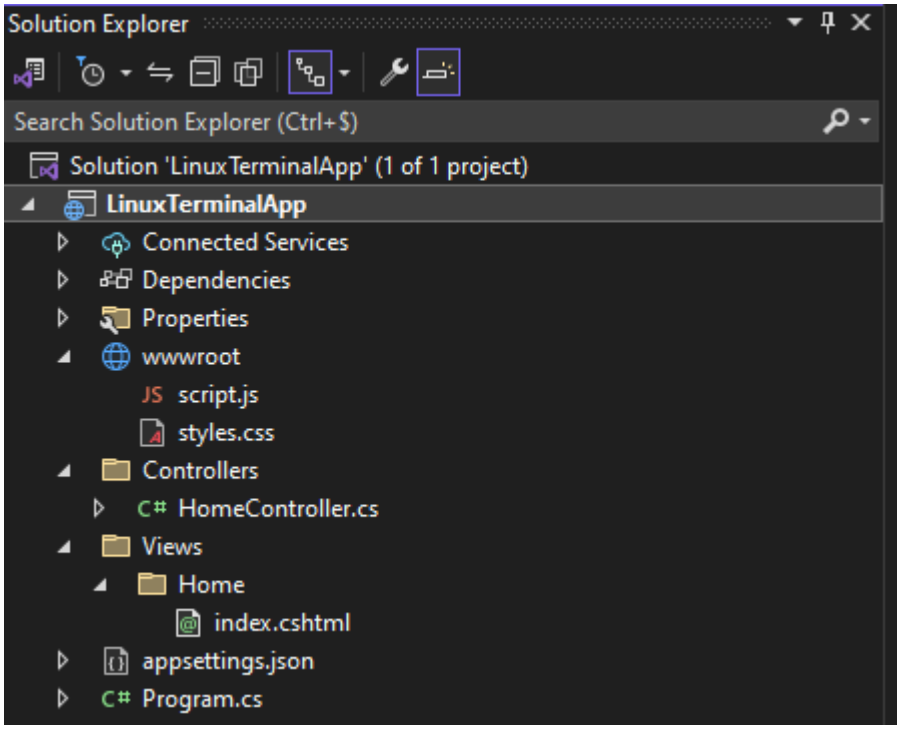

Figure 4.1: MVC folder structure.

2 Environment and development tools

In this section we briefly introduce all technologies and tools we used in our project .

2.1 Frameworks and libraries

in this part of chapter we will see all the frameworks and libraries we used in our web application

2.1.1 .NET

.NET has the following design points[13] [14]:
• Complete productivity with runtime, libraries, language, and tooling, all contributing to the developer user experience. Secure code is the primary computing model, while unsafe code allows for additional manual optimization[13] [14].

• Both static and dynamic code are supported, allowing for a wide range of distinct scenarios. Native code interoperability and hardware nativeness are low-cost and high-precision (raw API and instruction access)[13].

• Code is portable across platforms (OS and chip architectures), while platform targeting allows for specialization and optimization[13].

• Adaptability across programming domains (cloud, client, gaming) is possible through specialized implementations of a general-purpose programming model[13].

• Industry standards such as OpenTelemetry and gRPC are preferred over custom solutions[13].

.NET is maintained by Microsoft and the community. It is regularly updated to ensure users deploy secure and reliable applications in production[13] [14].

Figure 4.2: Microsoft .NET

2.1.2 ASP.NET Core

Millions of developers use or have used ASP.NET to build web applications. ASP.NET Core is a redesign of ASP.NET, including architectural changes that result in a lighter, more modular framework[15].

Figure 4.3: Asp.Net Core

ASP.NET Core provides the following benefits[15]:

- A unified story for building user interfaces and web APIs[15].
- Designed for testing[15].
- Razor pages make coding page-based scenarios easier and more efficient[15].
- Blazor lets you use C# in the browser with JavaScript. Shared client-side and server-side application logic, written entirely in .NET [15].
- Develop and run on Windows, macOS, and Linux[15].
- Open source and community-driven[15].
- Integrate modern client-side development frameworks and workflows[15].
- Support for hosting remote procedure calls (RPC) services using gRPC[15].
- A cloud-ready, environment-based configuration system[15].
- Dependency integration[15].
- A lightweight, efficient, modular HTTP request pipeline[15].
- Hosting capabilities on the following [15]:

-Kestrel
-IIS
-HTTP.sys
-Nginx
-Docker

- Parallel instances[15].
- Tools that simplify modern web development[15].

2.1.3 Razor Pages

Razor is a markup syntax for embedding .NET primarily based totally code into webpages. The Razor syntax includes Razor markup, C#, and HTML[16]. Files containing Razor typically have a .cshtml document extension. Razor is likewise discovered in Razor factor files (.razor)[16]. Razor syntax is just like the templating engines of numerous JavaScript single-web page application (SPA) frameworks, including Angular, React, VueJs, and Svelte[16].

2.2 Languages

this part of the chapter we introduce all the programming languages used in our web application , we can see the following figures from Github that shows statistics about used languages in our project :

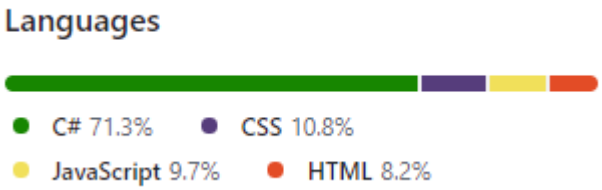

Figure 4.4: Languages used

More than 70% of the code is written in c#[17] , and mostly used for the back-end.

Figure 4.5: C#

We have 9.7% of the code written in JavaScript[18] which is the link between the front and the back-end.

Figure 4.6: JavaScript

2.3 IDE's

Visual Studio 2022

Visual Studio is an effective developer's software that you may use to finish the complete improvement cycle in a single place. It's a complete incorporated improvement environment (IDE) that you may use to write, edit, debug, and construct code. Then installation your app. Visual Studio consists of compilers, code crowning glory tools, supply control, extensions, and lots of different functions to beautify each level of the software program improvement process [19].

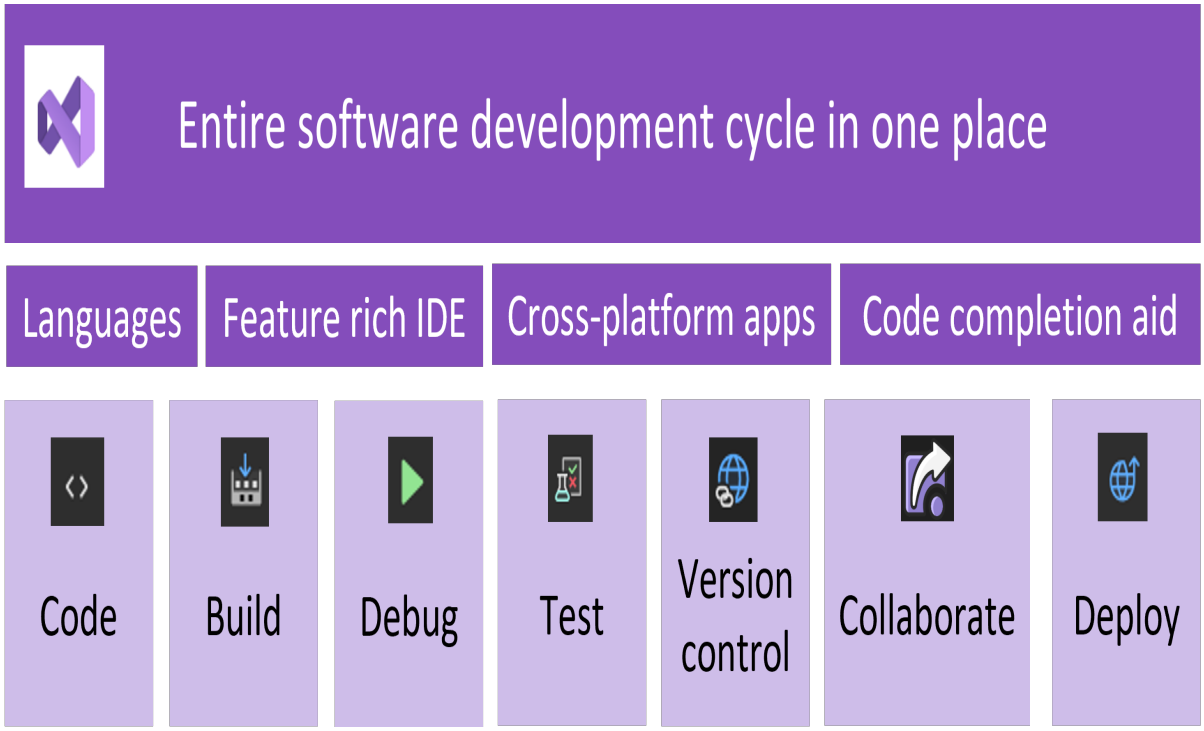

Figure 4.7: Visual Studio

With the kind of functions and languages assist in Visual Studio, you may develop from writing your first "Hello World" application to growing and deploying apps. For example, construct, debug, and test .NET and C++ apps, edit ASP.NET pages withinside the internet clothier view, expand cross-platform cellular and computing device apps with .NET, or construct responsive Web UIs in C#[19].

2.4 Server

In this part of chapter we see the tools that we used to emulate a server in a local computer.

2.4.1 Virtual box

VirtualBox is a effective x86 and AMD64/Intel64 virtualization product for agency in addition to domestic use. Not handiest is VirtualBox an incredibly function rich, excessive overall performance product for agency customers, it's also the handiest expert answer this is freely to be had as Open Source Software beneathneath the phrases of the GNU General Public License (GPL) model 3. See "About VirtualBox" for an introduction[20].

Presently, VirtualBox runs on Windows, Linux, macOS, and Solaris hosts and helps

a big wide variety of visitor running structures together with however now no longer restrained to Windows (NT 4.0, 2000, XP, Server 2003, Vista, 7, 8, Windows 10 and Windows 11), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x, 4.x, 5.x and 6.x), Solaris and OpenSolaris, OS/2, OpenBSD, NetBSD and FreeBSD[20].

VirtualBox is being actively advanced with common releases and has an ever developing listing of features, supported visitor running structures and systems it runs on.[20].

Figure 4.8: VirtualBox

2.4.2 Ubuntu Server

Ubuntu Server is a model of the Ubuntu running gadget designed and engineered as a spine for the internet[21].

Ubuntu Server brings financial and technical scalability in your datacentre, public or private. Whether you need to set up an OpenStack cloud, a Kubernetes cluster or a 50,000-node render farm, Ubuntu Server offers the nice cost scale-out overall performance available[21].

3 Application overview

In this section we explain how we implemented our project and how we used every tool and concept we talked about earlier in the previous chapters and sections.

3.1 code details

as we seen earlier the code is mostly written in c# which makes the back-end part of the application, and for the front-end we used HTML , CSS , and JavaScript.

Our project contain three main folders which they are :

• **wwwroot :** this folder contain all the CSS stylesheets which we call them in the HTML page, also this folder contains the JavaScript code which is responsible for the data extraction from the HTML page and calling the application controller to process that data and sends back the result which the JavaScript code is also responsible for displaying it in the HTML page , in our case it takes the Linux command that users enter, and the controller send it to the Linux server to executed, and as the controller receives the output it will be sent directly to the fetch API used by JavaScript to display it in the web page .

The figure 4.9 shows a pseudo code of the script written in JavaScript :

```
const input = document.getElementById('terminal');

input.addEventListener('keydown',function (event){
    if (event.key == 'Enter') {
        //commandline is the route of the controller
        var response = fetch("commandline/"+input.value);
        var data = await response.text();
    }
}
```

Figure 4.9: Script pseudo code

• **Controllers :** this is the most important folder in our web application , the responsibility of the files contained by this folder is processing each interaction the user makes with the application, in our case we have one controller that makes contact with the Linux server via SSH as soon as the command is received as a string from the the front-end, all of the code in this folder is written in C# .

The figure 4.10 shows a pseudo code of the controller written in C# :

```
Route["commandline"]
action (string command){
    return execute(command);
}
```

Figure 4.10: Controller pseudo code

• **Views :** This folder hold a simpler duty , mostly retrieving data or inputs from the user, and displaying the outputs received by the controller, in our case we used razor pages , written HTML and C# .

The figure 4.12 shows a pseudo code of the view written in HTML :

```
<html>
    <head>
        <title>Linux terminal emulator</title>
    </head>
    <body>
        <terminal_emulator id="terminal"> </terminal_emulator>
    </body>
</html>
```

Figure 4.11: View pseudo code

The source code is included in the Github platform , you can scan the link in the next figure 4.12 (https://github.com/BghAek/LinuxEmulator/tree/main/LinuxTerminalApp) :

Figure 4.12: Source Code

Chapter 5

Conclusion & Future Work

In this chapter we discuss what we have already accomplished concerning this project and state all the problems that we encountered, and what should be done as a mandatory work so the application can be usable by students, and also what tools we need to get that work done, and also what can be done as an enhancement or upgrade to the project .

1 Problem statement

As far as we accomplished in this project, our web application is still in the staging phase and not ready to be launched and used by multiple students, however we stated the main problems and later on the next section we discuss the solutions to be done in the future.

First problem : In our case of development we used a single virtual machine as a Linux server for the purpose of testing, however using the same approach in the real world will not work very well, because when multiple users access the application they all access the same virtual machine and they can corrupt the system for each other, so the objective here is to have for each new user his own Linux server session where he can practice any command he wants without corrupting the system for the other users.

Second problem : Since we allow user to have "sudo" privilege , that will make a massive security issue if the user can elevate his access rights to "root" privilege, so the objective here is to implement a mechanism that makes the application safe so malicious users cannot corrupt the system.

2 Solutions & Future work

In this section we discuss the solutions for the two main problems in the application and what are the tools we need to accomplish our result, and also suggest extra future work for for the purpose of enhancing our application.

The first solution comes to mind concerning the two main problems we mentioned earlier in this chapter is Containerization, Containers (e.g., Docker) are a great way to provide isolated environments for each user. Each user would get their own container, which ensures they don't interfere with each other.

Why use Docker ?
With Docker, you can quickly deploy and scale applications in any environment and know your code will run[22] [23].

Figure 5.1: Docker

How to use Docker in our Application ?

So basically instead of using using a single virtual machine that directly execute the commands, we implement a Docker image of a Linux server that resides in the in a VPS , and each time a new user access the application, a new container of that Linux server image gets generated only for that specific user, and as soon as the user leaves, the container gets dismissed and release all the hardware resources (CPU , RAM) it was using so it can be used again.

After using Containerization approach we solve both of the problems we encountered, we separate each user from the other, so every user can have his own container or session, and for the security there will not be problems anymore because a user can only harm his own specific container, he can even run a "sudo shutdown" command and that will only shut down his own container .

Extra future work :

This part can be considered as an upgrade to the project, which is implementing a database for the application, where each user has his own account, and record each command the user enters into the Linux terminal emulator, so every time the user access his account and a new container of a Linux server gets generated, we make a code in the back-end that execute every and each command the user practiced the last time he was logged in by order, so he can pick up learning a and practicing where he left off the last time he used the application.

3 Conclusion

So in the end we have an open source tool that comes as a web application, that emulates a Linux terminal, that serve the same purpose with the other tools, with unlimited privilege and access rights for the user and less limitations, so students can practice any linux command in a safe environment without having the risk to try it in their own system with probability of damaging it , and we also invite all the students of the university of Amar Telidji that are interested in learning linux to download the code from the github link and they can test and try our application and help to upgrade and work on it , so it can be ready to be launched and hosted in a real server and used by many student .

References

[1] D. J. Barrett. Linux pocket guide. 2004. ISBN 9780596006280. 2

[2] What is linux? URL https://www.linux.com/what-is-linux/. Accessed: 28-08-2024. 2

[3] W. E. Shotts. The linux command line : a complete introduction. San Francisco: No Starch Press, 2012. ISBN 9781593273897. 3

[4] OccupyTheWeb. Linux basics for hackers : getting started with networking, scripting, and security in kali. San Francisco: No Starch Press, 2019. ISBN 1593278551. 3

[5] URL https://cocalc.com/features/terminal. Accessed: 28-08-2024. 4

[6] URL https://www.tutorialspoint.com/codingground.htm. Accessed: 28-08-2024. 7

[7] Basics of clean architecture with c. URL https://www.macrix.eu/en/basics-of-clean-architecture-with-c/#). Accessed: 28-08-2024. 13, 14

[8] Robert C. Martin Series. Clean architecture: A craftsman's guide to software structure and design. pages 132665–132676, 2017. 13

[9] Steve Smith. Overview of asp.net core mvc. URL https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-website). Accessed: 28-08-2024. 15

[10] Adam Freeman. Pro asp.net mvc 5. 2013. ISBN 9781430265290. 16

[11] M. Seidl. Uml @ classroom : an introduction to object-oriented modeling. Cham: Springer, 2015. ISBN 9783319127415. 16

[12] URL https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/). Accessed: 28-08-2024. 16

[13] Introduction to .net. URL https://learn.microsoft.com/en-us/dotnet/core/introduction). Accessed: 28-08-2024. 20

[14] D. Metzgar. .net core in action. Simon and Schuster, 2013. ISBN 9781617294617. 20

[15] Shaun Luttin Daniel Roth, Rick Anderson. Overview of asp.net core. URL https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0). Accessed: 28-08-2024. 21, 22

[16] Dan Vicarel Rick Anderson, Taylor Mullen. Razor syntax reference for asp.net core. URL https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor?view=aspnetcore-8.0. Accessed: 28-08-2024. 22

[17] A. Troelsen and P. Japikse. Pro c 9 with .net 5 : foundational principles and practices in programming. Berkeley, Ca: Apress L. P., . Copyright, 2021. ISBN 9781484278680. 22

[18] D. Crockford. Javascript : the good parts. Farnham: O'reilly, 2008. ISBN 0596517742. 23

[19] What is visual studio? URL https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022. Accessed: 28-08-2024. 23, 24

[20] URL https://www.virtualbox.org/. Accessed: 28-08-2024. 24, 25

[21] Ubuntu server documentation. URL https://ubuntu.com/server/docs. Accessed: 28-08-2024. 25

[22] URL https://aws.amazon.com/docker/. Accessed: 28-08-2024. 30

[23] N. Poulton. Docker deep dive : zero to docker in a single book. Germany: Nigel Poultons, 2020. ISBN 9781521822807. 30