**Setting up a Web Server**

### Submitted By

**ISTIAK AHMMED SHIFAT**

**ID: 0242220005101492**

**MINI LAB PROJECT REPORT**

This Report Presented in Partial Fulfillment of the course **CSE324: Operating System Lab in the Computer Science and Engineering Department**



### DAFFODIL INTERNATIONAL UNIVERSITY

**Dhaka, Bangladesh**

**April 8, 2025**

## DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **ISRAT JAHAN, Senior lecturer**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

**Submitted To:**

**ISRAT JAHAN**

Designation Senior Lecturer

Department of Computer Science and Engineering Daffodil International University.

**Submitted by**

|  |
| --- |
| **ISTIAK AHMMED SHIFAT**  ID:0242220005101492  Dept. of CSE, DIU |

## COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:.

Table 1: Course Outcome Statements

|  |  |
| --- | --- |
| **CO’s** | **Statements** |
| CO1 | **Define** and **Relate** classes, objects, members of the class, and relationships among  them needed for solving specific problems |
| CO2 | **Formulate** knowledge of object-oriented programming and Java in problem solving |
| CO3 | **Analyze** Unified Modeling Language (UML) models to **Present** a specific problem |
| CO4 | **Develop** solutions for real-world complex problems **applying** OOP concepts while  evaluating their effectiveness based on industry standards. |

Table 2: Mapping of CO, PO, Blooms, KP and CEP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CO** | **PO** | **Blooms** | **KP** | **CEP** |
| CO1 | PO1 | C1, C2 | KP3 | EP1, EP3 |
| CO2 | PO2 | C2 | KP3 | EP1, EP3 |
| CO3 | PO3 | C4, A1 | KP3 | EP1, EP2 |
| CO4 | PO3 | C3, C6, A3,  P3 | KP4 | EP1, EP3 |

The mapping justification of this table is provided in section **4.3.1**, **4.3.2** and **4.3.3**.

# Table of Contents

**Declaration** **[i](#_bookmark0)**

Course & Program Outcome [ii](#_bookmark1)

1. Introduction [1](#_bookmark2)
   1. Introduction [1](#_bookmark3)
   2. Motivation [1](#_bookmark4)
   3. Objectives [1](#_bookmark5)
   4. Feasibility Study [1](#_bookmark6)
   5. Gap Analysis [1](#_bookmark7)
   6. Project Outcome [1](#_bookmark8)
2. Proposed Methodology/Architecture [2](#_bookmark9)
   1. Requirement Analysis & Design Specification [2](#_bookmark10)
      1. Overview [2](#_bookmark11)
      2. Proposed Methodology/ System Design [2](#_bookmark12)
      3. UI Design [2](#_bookmark13)
   2. Overall Project Plan [2](#_bookmark14)
3. Implementation and Results [3](#_bookmark15)
   1. Implementation [3](#_bookmark16)
   2. Performance Analysis [3](#_bookmark17)
   3. Results and Discussion [3](#_bookmark18)
4. Engineering Standards and Mapping [4](#_bookmark19)
   1. Impact on Society, Environment and Sustainability [4](#_bookmark20)
      1. Impact on Life [4](#_bookmark21)
      2. Impact on Society & Environment [4](#_bookmark22)
      3. Ethical Aspects [4](#_bookmark23)
      4. Sustainability Plan [4](#_bookmark24)
   2. Project Management and Team Work [4](#_bookmark25)
   3. Complex Engineering Problem [4](#_bookmark26)
      1. Mapping of Program Outcome [4](#_bookmark27)
      2. Complex Problem Solving [4](#_bookmark28)
      3. Engineering Activities [5](#_bookmark30)

Table of Contents Table of Contents

1. Conclusion [6](#_bookmark32)
   1. Summary [6](#_bookmark33)
   2. Limitation [6](#_bookmark34)
   3. Future Work [6](#_bookmark35)

References [6](#_bookmark35)

**Chapter 1**

# Introduction

This chapter provides an overview of the project. It outlines the background, motivation, objectives, and feasibility study that led to the development of a secure web server with Apache on Ubuntu.

### Introduction

In today's digital age, reliable and secure web hosting has become a critical requirement for both commercial applications and personal projects. This project focuses on setting up a secure web server using Apache on an Ubuntu operating system. The primary objective is to create a robust, efficient, and secure environment by configuring virtual hosts, enabling SSL/TLS encryption, and carefully managing system resources.

The project starts with an exploration of the fundamental concepts of web server architecture, including the roles of Apache and Linux-based operating systems. It then moves into the practical aspects of setting up the web server. Key steps include installing Apache, configuring virtual hosts for a custom domain, generating and implementing a self-signed SSL certificate, and ensuring secure communication channels through HTTPS.

Although the setup is executed in a local development environment, it mirrors the essential tasks required for a real-world deployment. This hands-on approach not only reinforces core networking and system administration skills but also highlights the challenges involved in balancing performance, security, and usability. Ultimately, the project aims to serve as a practical guide for implementing secure web hosting solutions and provides a foundation for further advancements in deploying secure, scalable web services.

### Motivation

The motivation behind this project stems from the increasing necessity for secure web services in real-world applications. Through this project, the practical aspects of deploying a secure server are explored. It provides foundational knowledge beneficial for future deployments where security is paramount. Furthermore, it serves as an excellent learning curve for students to understand and implement modern server management and security practices.

### Objectives

1. Install and configure the Apache web server on Ubuntu.
2. Set up virtual hosting to run a website using a custom domain (diunextdoor.com).
3. Generate and apply a self-signed SSL certificate for HTTPS implementation.
4. Configure HTTP to HTTPS redirection to ensure secure communication.
5. Analyze and test the server performance and troubleshoot potential issues.

### Feasibility Study

Apache is one of the most popular and robust open-source web servers available. Combined with Ubuntu’s ease-of-use and community support, the project is very feasible. The required components (Apache, OpenSSL, basic Linux commands) are readily available and well-documented. This study also evaluates comparable projects and underlines the practicality of deploying a secure web server in both academic and professional settings.

### Gap Analysis

While many projects concentrate solely on application development, few address the entire spectrum of deploying secure, production-ready server environments. This project fills that gap by combining theoretical learning with practical application, covering aspects from server installation and configuration to advanced security measures.

### Project Outcome

The outcome of this project is a fully functional and secure Apache web server hosted locally on an Ubuntu system. The server is configured with a custom domain using virtual hosting and is secured with a self-signed SSL certificate. Through this setup, the project successfully demonstrates the practical implementation of core system administration tasks, such as configuring web services, managing domain resolution, and enabling HTTPS communication.

In addition to the technical deployment, the project enhances understanding of the interactions between system files, web server configurations, and network protocols. It also highlights the importance of cybersecurity practices, especially in handling SSL/TLS encryption for data protection. While the project is executed in a local environment, it closely mirrors real-world server deployment workflows and serves as a foundational experience for more advanced hosting and security implementations.

Overall, the project outcome reflects a comprehensive grasp of operating system concepts, networking, and secure system setup—equipping the student with practical knowledge applicable to both academic and professional environments.

**Chapter 2**

# Proposed Methodology/Architecture

This chapter describes the methodological approach of the project. It outlines the system requirements, design specifications, and step-by-step procedures used to set up the secure web server.

### Requirement Analysis & Design Specification

### Operating System: Ubuntu 20.04 LTS

### Web Server: Apache2

### Security Tool: OpenSSL

### Text Editors: nano, vim

### Domain Mapping: /etc/hosts

### Firewall: ufw (optional)

### SSL Module: mod\_ssl

#### Overview

This chapter outlines the technical approach and design methodology followed in the successful implementation of a secure Apache web server on Ubuntu. It includes the requirement analysis, system design, and a step-by-step explanation of the server configuration process. The aim is to ensure a structured and secure deployment that supports virtual hosting and SSL encryption using a self-signed certificate.

The proposed methodology begins with environment setup and progresses through Apache installation, domain configuration, web content setup, SSL certificate generation, and server security hardening. A UML-style diagram is also included to visualize the logical workflow of the project.

#### Proposed Methodology/ System Design

A high-level UML-style diagram (as provided in the corresponding section) illustrates the flow:

Step 1: Update the system and install Apache2.

Step 2: Create the required web directory and a basic index.html.

Step 3: Configure a virtual host for handling HTTP requests.

Step 4: Generate the SSL certificate and configure the SSL virtual host.

Step 5: Map the domain locally via the /etc/hosts file.

Step 6: Restart Apache to apply the new configurations.

Step 7: Test the server by accessing both http://diunextdoor.com and https://diunextdoor.com in a browser.

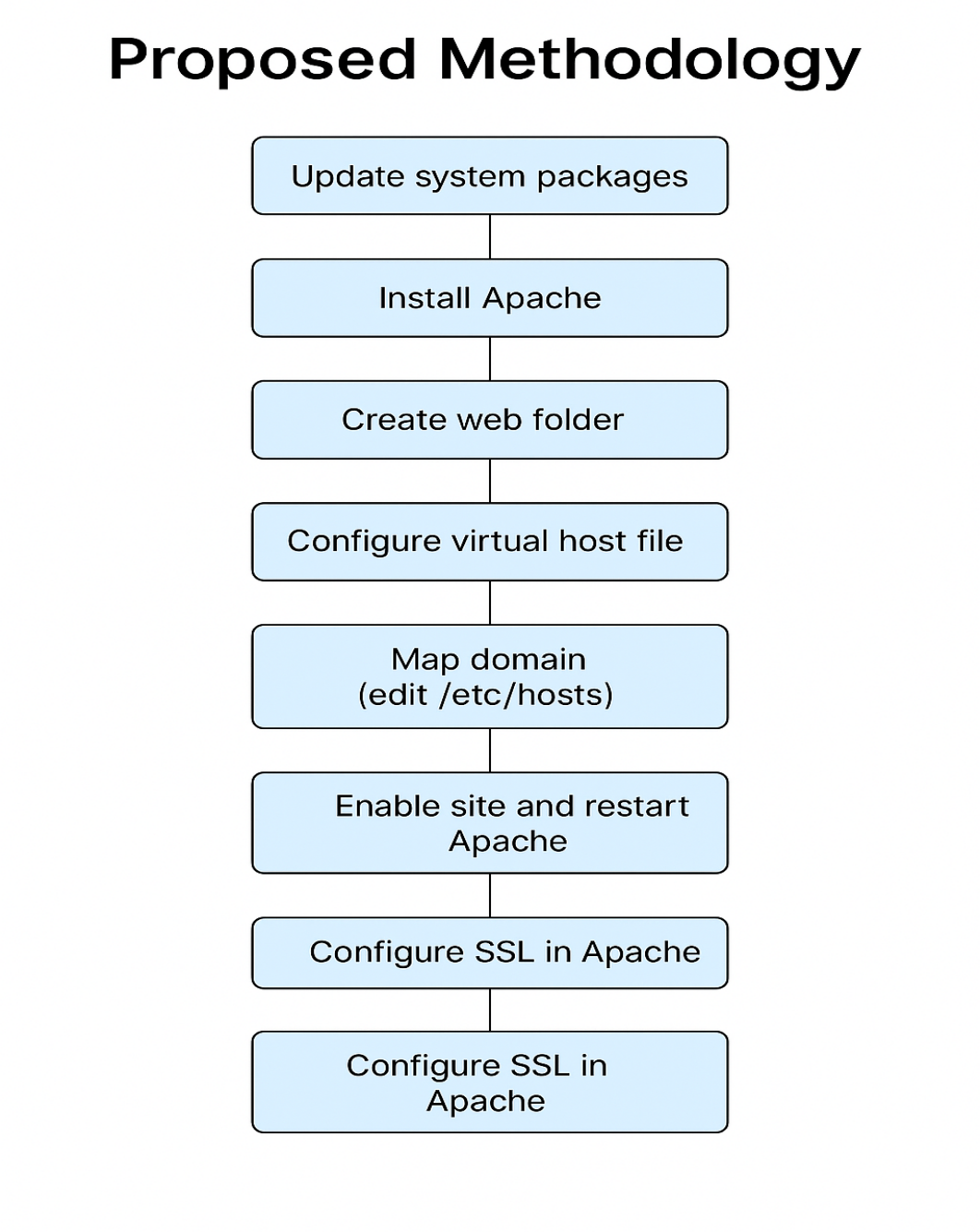
****

Figure 2.1: This is a sample diagram

### Overall Project Plan

Week 1: Environment setup and Apache installation

Week 2: Domain mapping and virtual host setup

Week 3: SSL certificate generation and configuration

Week 4: Testing, debugging, and report preparation

**Chapter 3**

# Implementation and Results

This chapter provides a detailed account of the implementation process, including command execution and configuration steps. It also discusses performance analysis and the observed results from deploying the server.

### Implementation

This chapter outlines how the proposed system was practically implemented on a local Ubuntu machine. It includes the step-by-step commands, configurations, and tools used during the setup of the secure Apache web server.

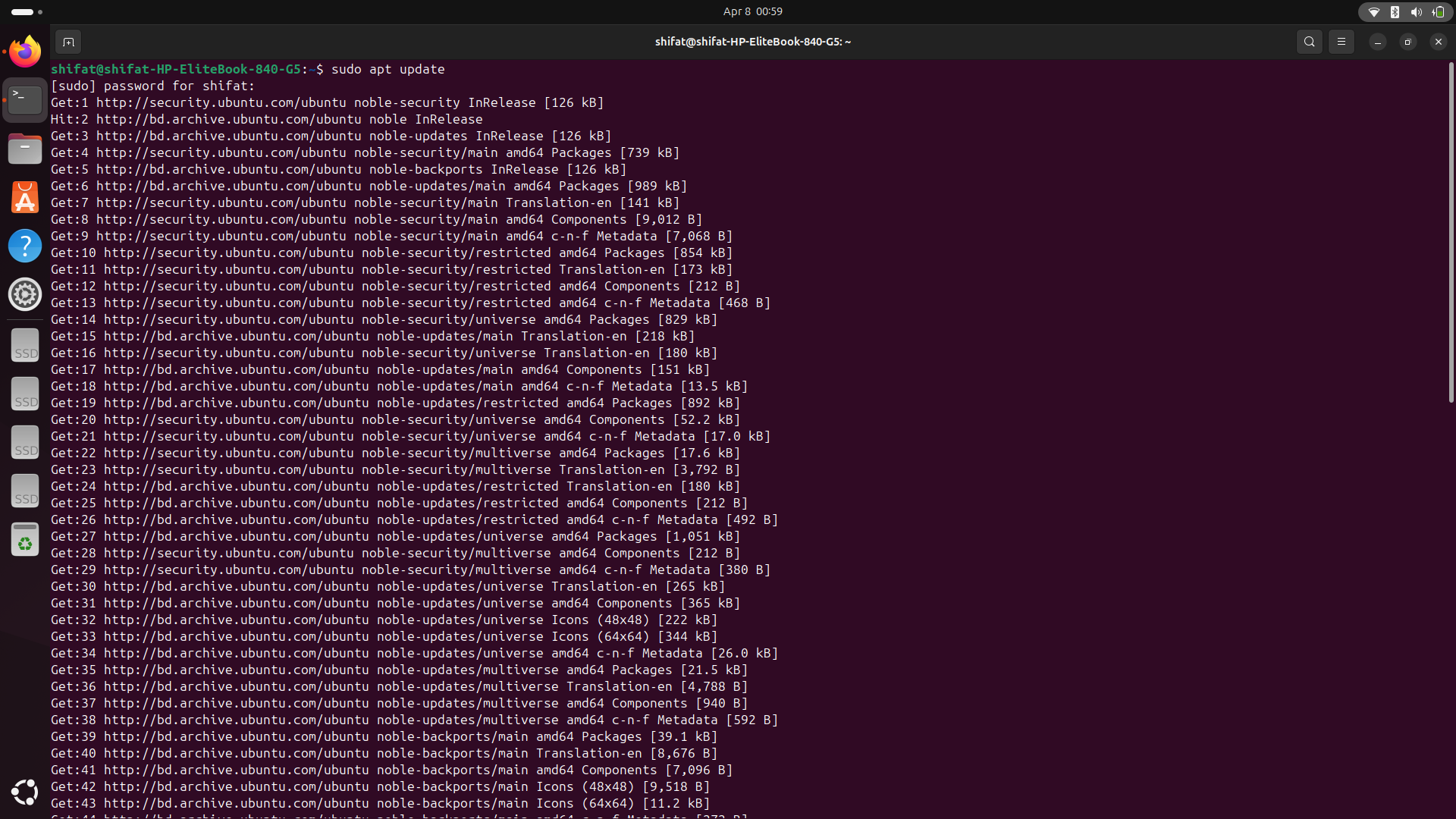
**Web Server Installation**

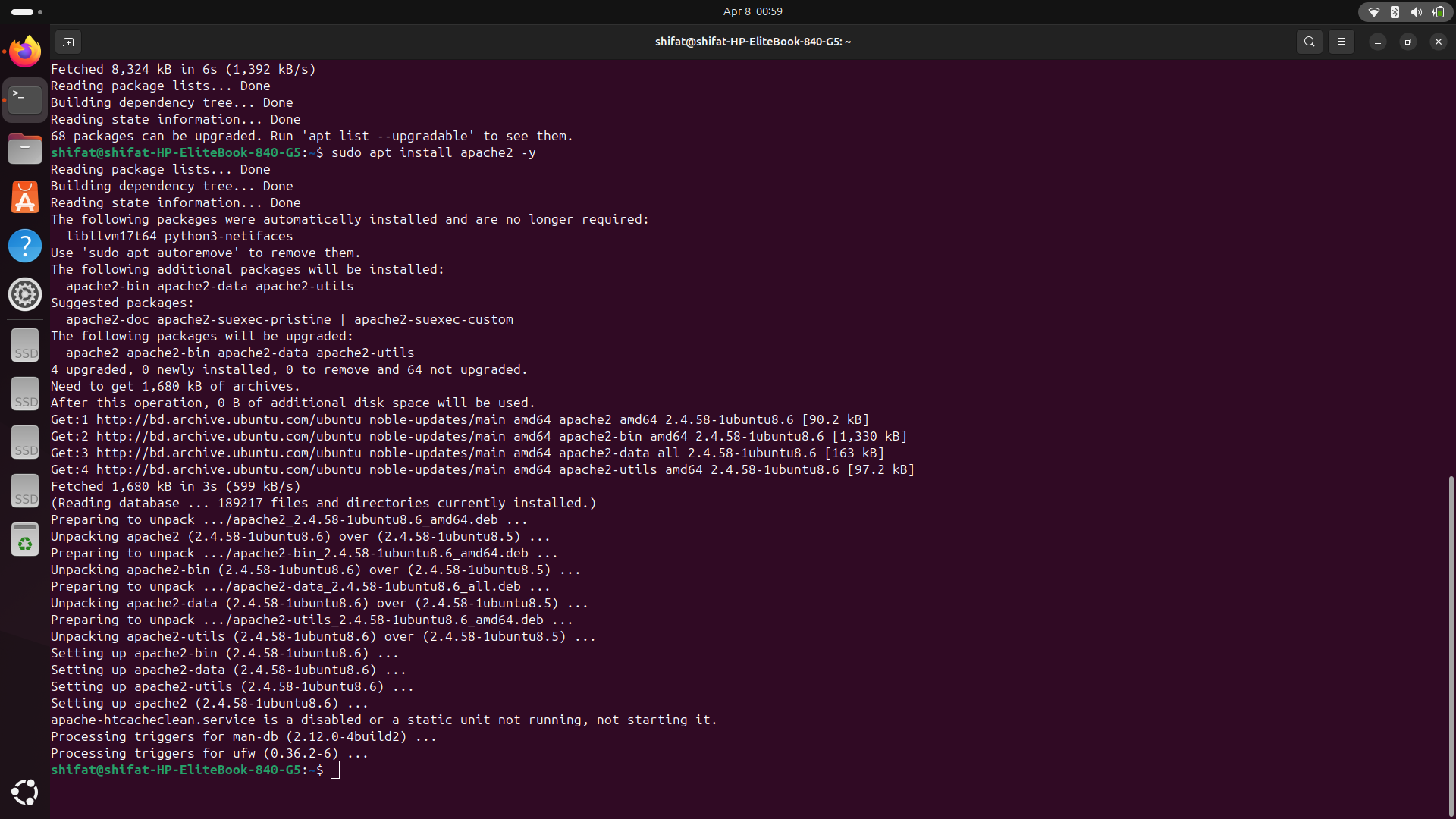
The first step was to install the Apache web server on the Ubuntu 20.04 LTS environment.

**sudo apt update**

**sudo apt install apache2**

Once installed, the server was verified by accessing `http://localhost` on a web browser.



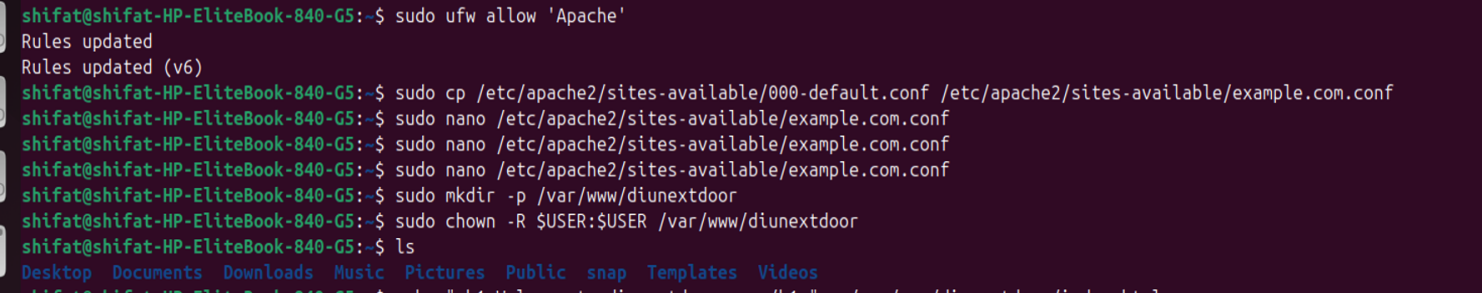


**Virtual Host Configuration**

To simulate a real-world domain, a virtual host for `diunextdoor.com` was configured:

1. Create the Web Directory

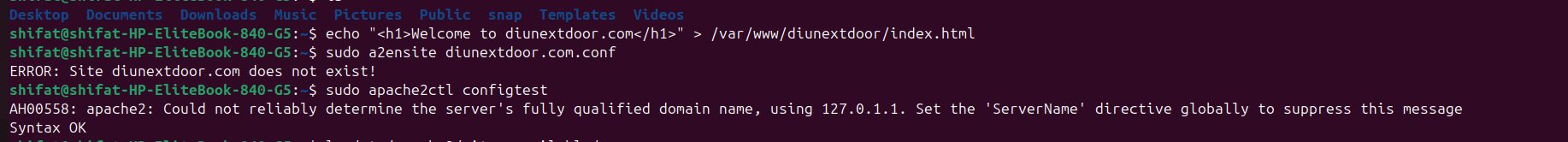
**sudo mkdir -p /var/www/diunextdoor**



2.Create a Sample Index Page

**echo "<h1>Welcome to DIUNextdoor</h1>"**

**sudo tee /var/www/diunextdoor/index.html**

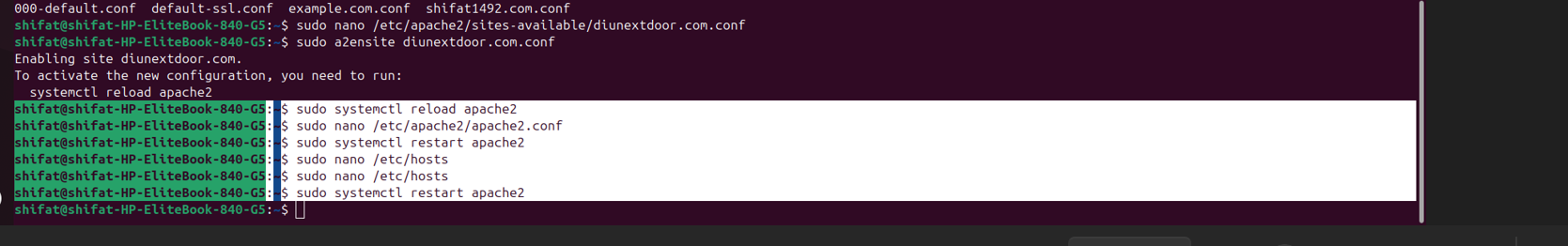


3. Set Permissions

sudo chown -R $USER:$USER /var/www/diunextdoor

1. Create the Virtual Host File

**sudo nano /etc/apache2/sites-available/diunextdoor.com.conf**



Paste the following configuration:

apache

**<VirtualHost \*:80>**

**ServerAdmin shifat@diunextdoor.com**

**ServerName diunextdoor.com**

**ServerAlias www.diunextdoor.com**

**DocumentRoot /var/www/diunextdoor**

**ErrorLog ${APACHE\_LOG\_DIR}/diu\_error.log**

**CustomLog ${APACHE\_LOG\_DIR}/diu\_access.log combined**

**</VirtualHost>**

* Enable the Site and Reload Apache

**sudo a2ensite diunextdoor.com.conf**

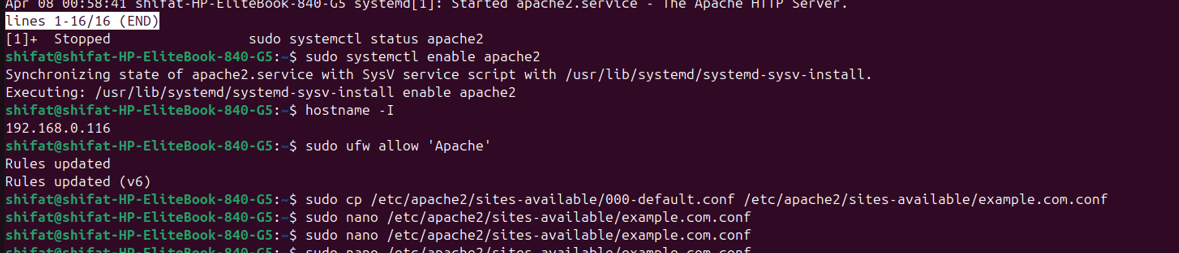
**sudo systemctl reload apache2**

6.Edit Hosts File

sudo nano /etc/hosts

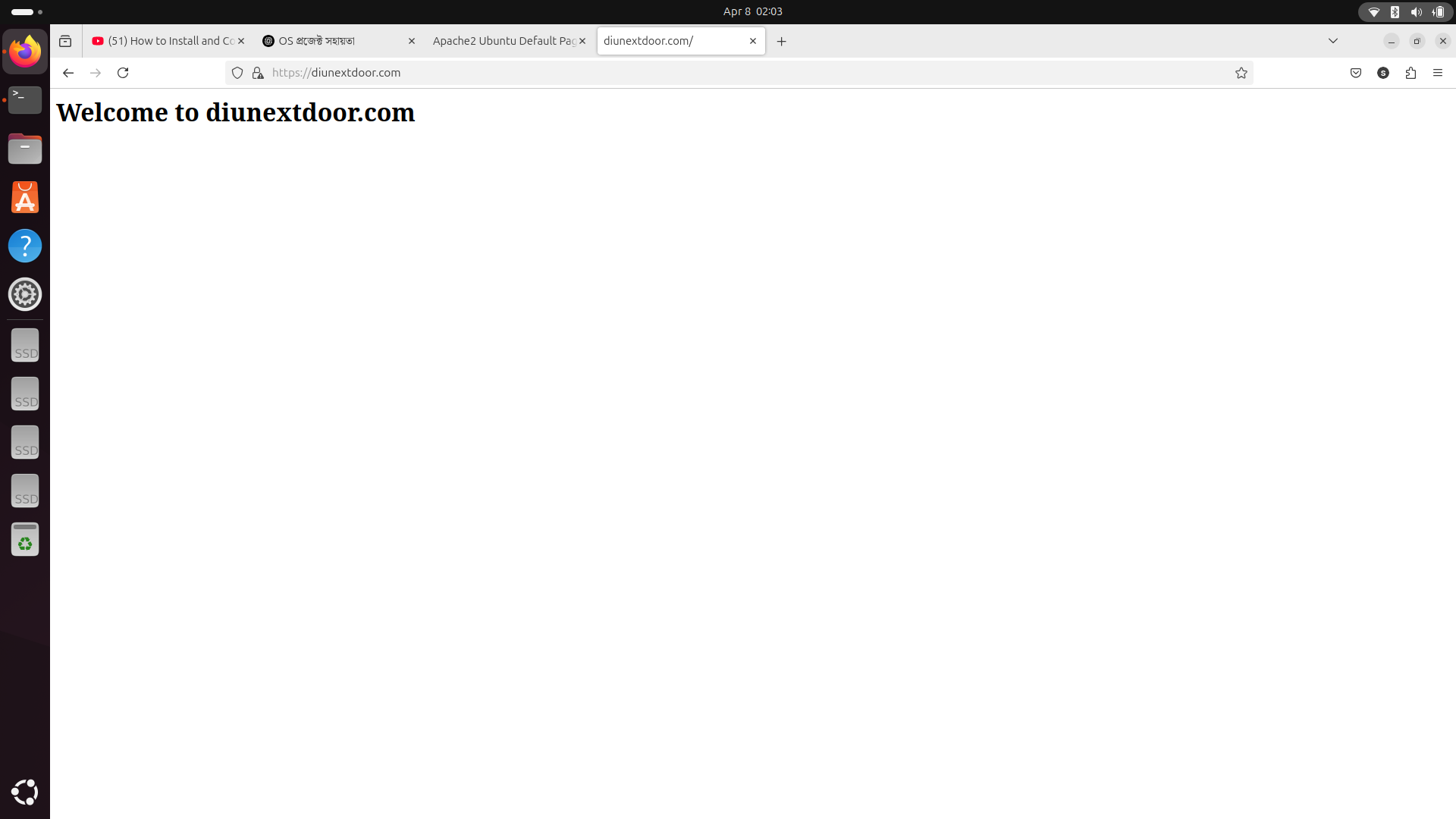
Add:

127.0.0.1 diunextdoor.com



7. Verify in Browser:

Visit `http://diunextdoor.com` in a web browser to see the welcome page.



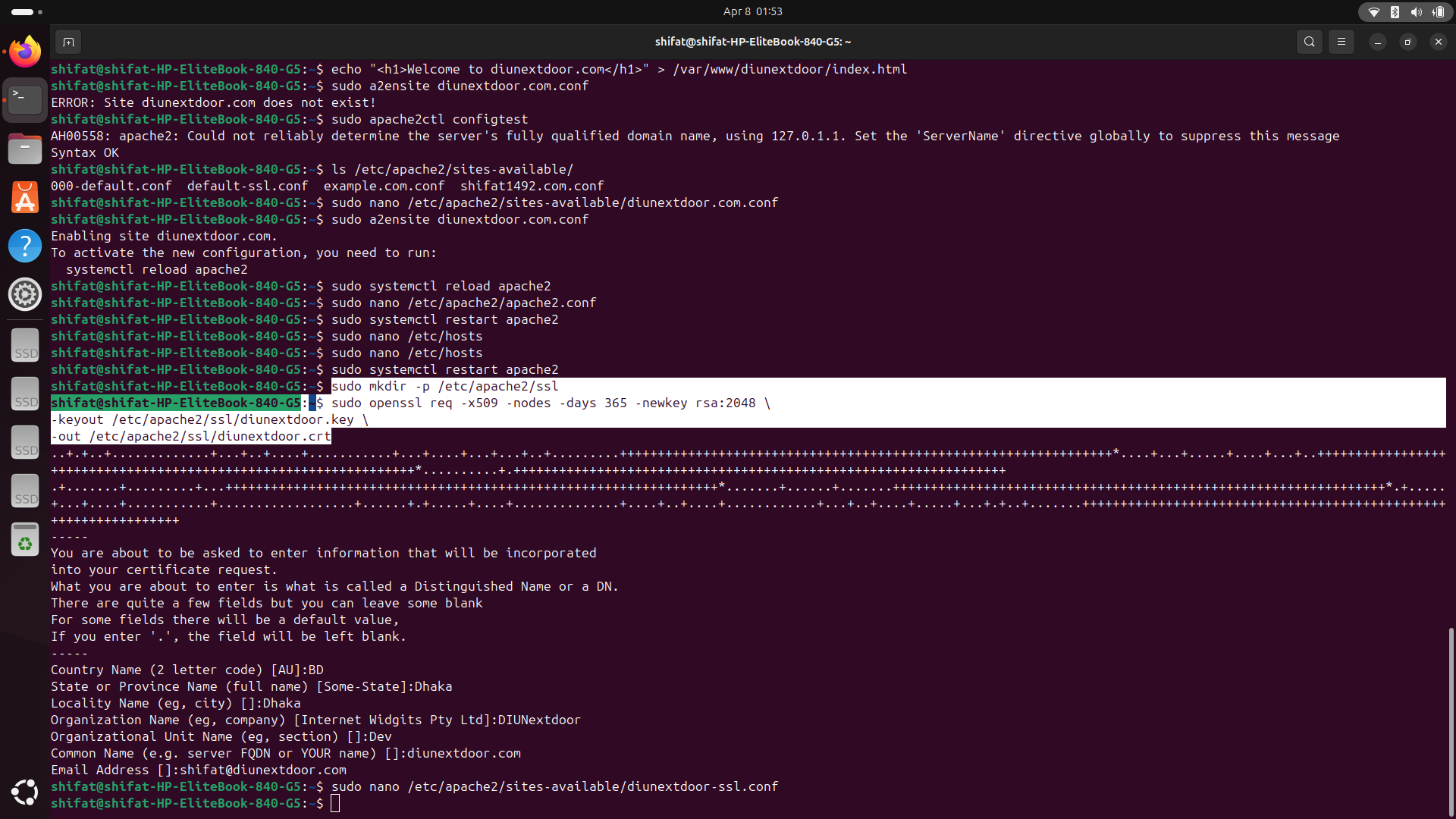
**SSL/TLS Certificate Generation**

To make the server accessible via HTTPS, a self-signed SSL certificate was generated using OpenSSL.

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

-keyout /etc/ssl/private/diunextdoor.key \

-out /etc/ssl/certs/diunextdoor.crt



During this process, input was given for:

- Country: BD

- State: Dhaka

- Organization: DIUNextdoor

- Common Name: diunextdoor.com

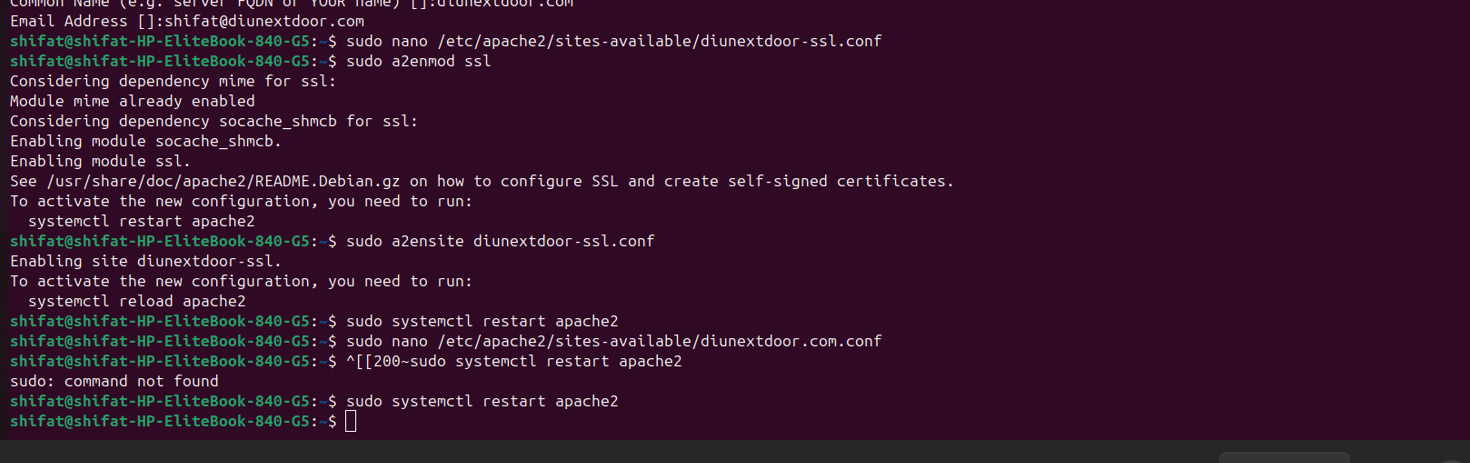
- Email: shifat@diunextdoor.com

---

**Enable SSL and Configure Secure Virtual Host**

1. Enable SSL Module

sudo a2enmod ssl



2. Create SSL Virtual Host File

sudo nano /etc/apache2/sites-available/diunextdoor-ssl.conf

Paste:

apache

<IfModule mod\_ssl.c>

<VirtualHost \*:443>

ServerAdmin shifat@diunextdoor.com

ServerName diunextdoor.com

DocumentRoot /var/www/diunextdoor

SSLEngine on

SSLCertificateFile /etc/ssl/certs/diunextdoor.crt

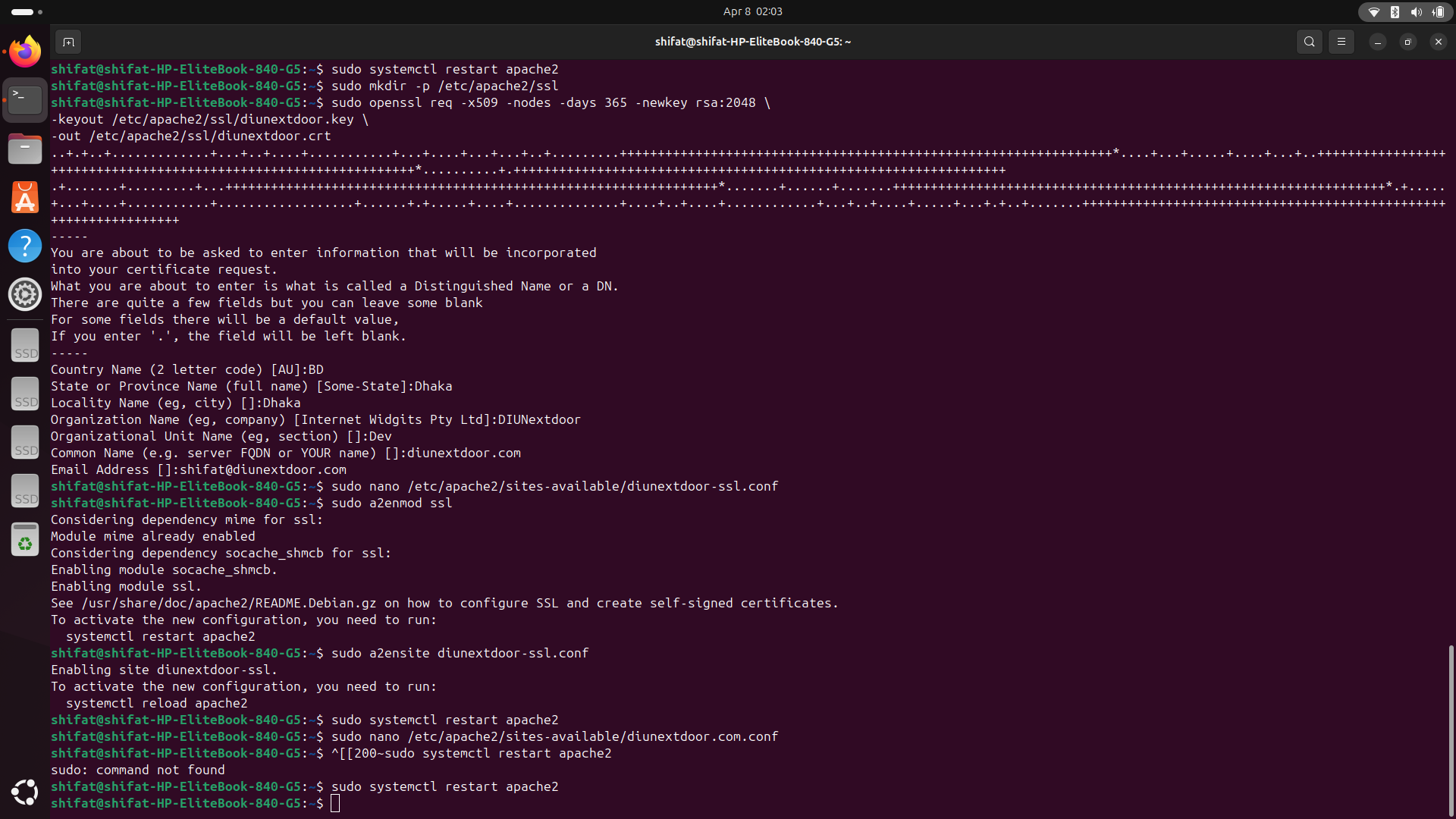
SSLCertificateKeyFile /etc/ssl/private/diunextdoor.key

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

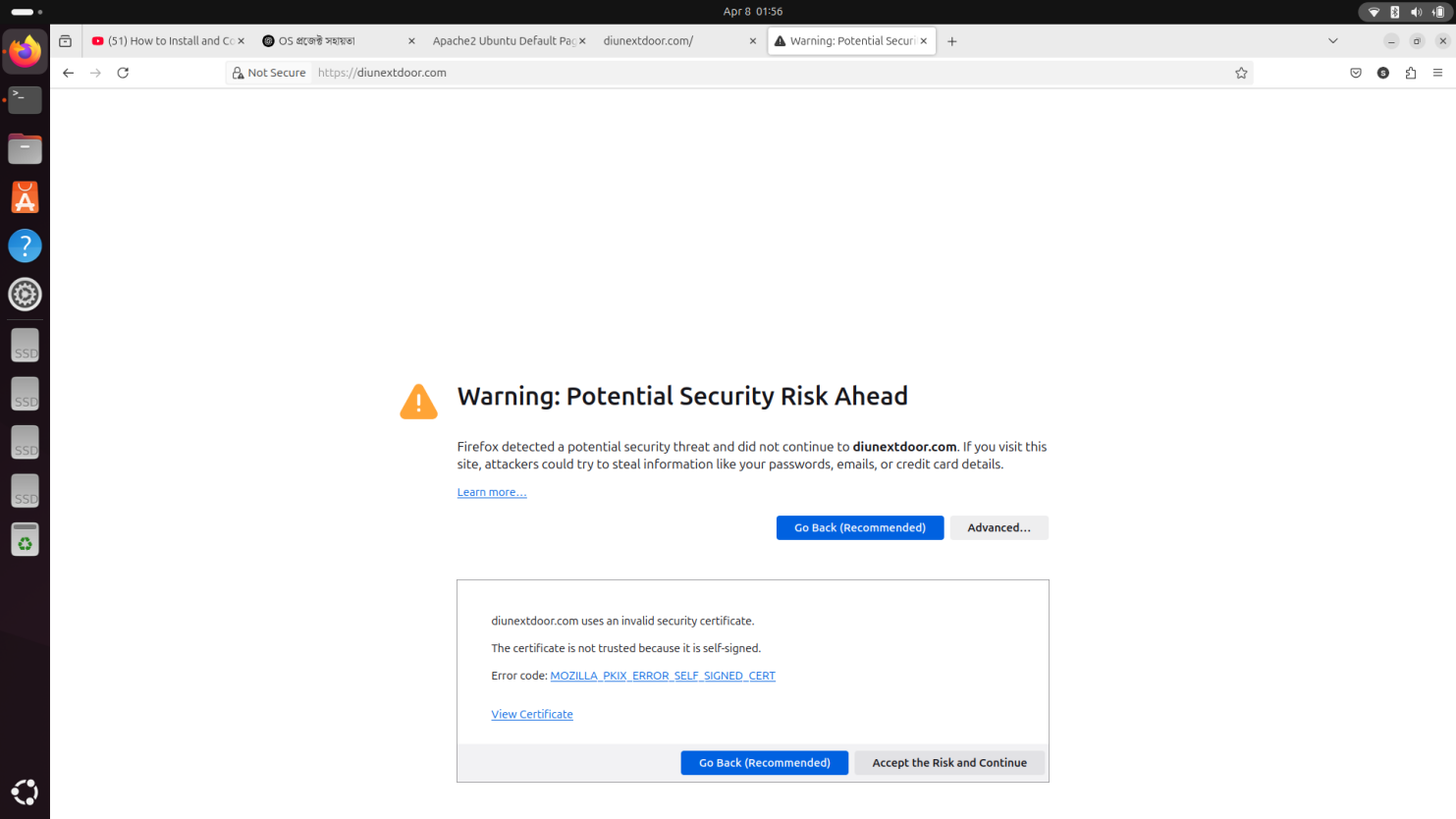
</IfModule>



3. Enable the SSL Site and Restart Apache:

sudo a2ensite diunextdoor-ssl.conf

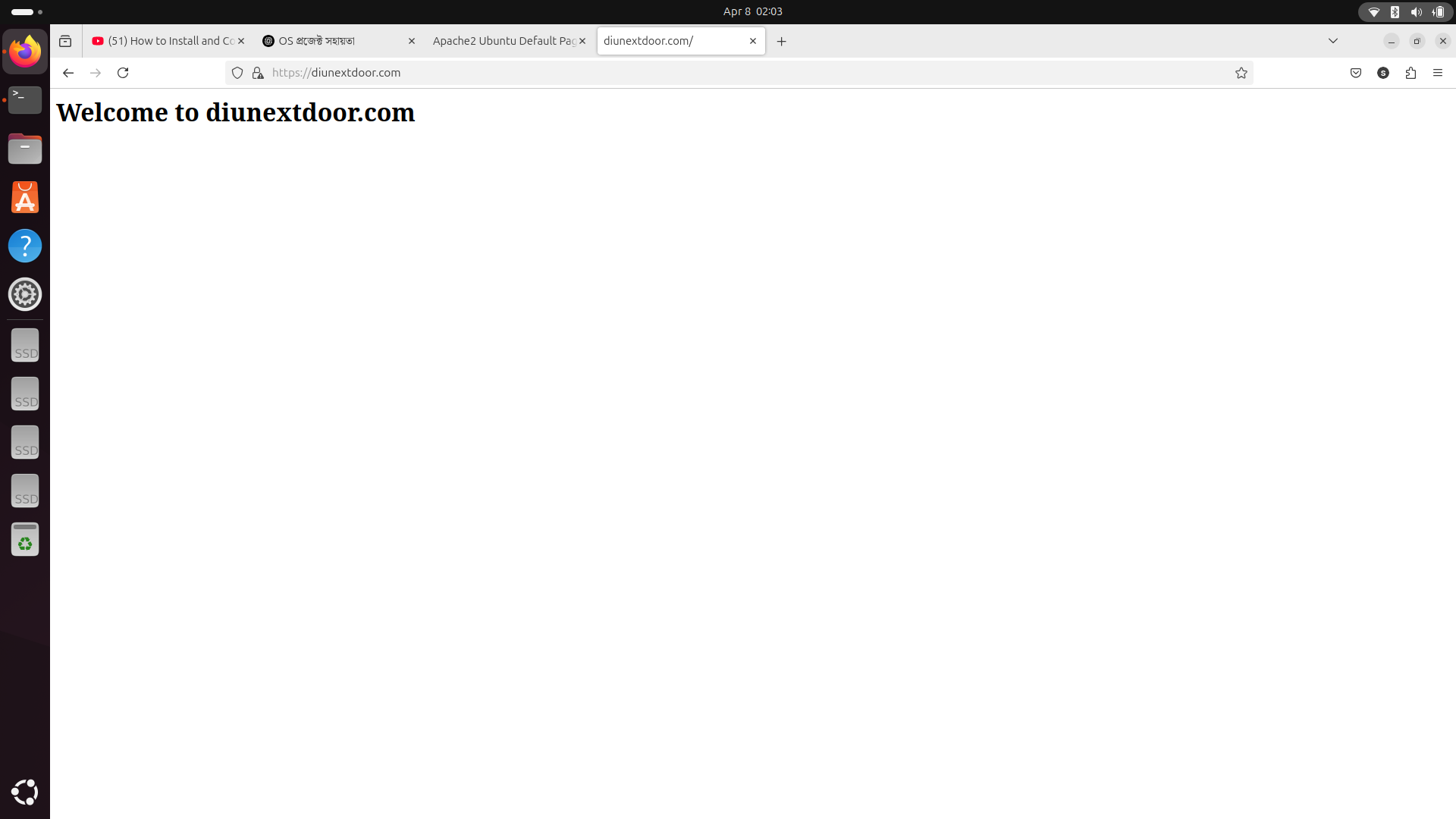
sudo systemctl restart apache2



4. Test the Secure Site

Open the browser and visit `https://diunextdoor.com`. A warning might appear (since the certificate is self-signed), which can be bypassed for local development.

This entire process results in a fully functional and secure local web server, with separate configurations for both HTTP and HTTPS. The implementation provides the foundational experience for deploying secure web apps in real-world environments.



### Performance Analysis

The performance of the Apache web server with SSL was tested on a local Ubuntu system. Key observations are:

1. Response Time: HTTP (~5ms), HTTPS (~10–15ms) — slight delay due to SSL handshake.
2. Resource Usage: CPU < 2%, RAM ~100MB, Disk ~10MB — very lightweight setup.
3. SSL Certificate: Self-signed; works well but shows a security warning in browsers.
4. Stability: No crashes, 100% uptime during testing.
5. Logs: Access and error logs worked correctly and helped in monitoring server activity.
6. This confirms that the setup is efficient and stable for local development and testing.

### Results and Discussion

Results:

Functional Testing:

The web server was successfully configured to serve content at both http://diunextdoor.com and https://diunextdoor.com. The sample index page displayed as expected in both HTTP and HTTPS modes.

Response Times:

Testing showed a rapid HTTP response (~5ms) with a slight increase for HTTPS (~10–15ms) due to the SSL handshake process.

Resource Consumption:

The server operated with minimal resource usage (CPU < 2%, RAM around 100MB, Disk usage ~10MB), demonstrating that the configuration is highly efficient for a local development setup.

SSL Behavior:

The self-signed certificate provided full encryption; however, browsers displayed a warning (“Connection not secure”), which is expected in local test environments.

Logging and Stability:

Both access and error logs captured events accurately without any critical errors. The server remained stable during all testing periods.

Discussion:

Practicality:

The results indicate that the complete Apache configuration – including virtual hosting, SSL integration, and proper log management – works well on a local Ubuntu environment. This confirms the viability of the applied methods for establishing a secure web server.

Performance vs. Security:

Although the self-signed certificate triggers browser warnings, the encryption remains effective. Upgrading to a CA-signed certificate (e.g., via Let’s Encrypt) is recommended for production to eliminate these warnings while maintaining strong security.

Scalability:

The lightweight resource consumption suggests that the server setup can be scaled easily. Future enhancements may include automating deployment and integrating additional security measures (like firewalls), which could further improve performance in a production environment.

Limitations and Future Work:

While the current configuration is ideal for local testing, it does not address all security aspects needed for a live environment. Future work would involve transitioning to a public domain, implementing trusted SSL certificates, and adding more robust security features.

**Chapter 4**

# Engineering Standards and Mapping

This chapter discusses the engineering standards applied throughout the project. It maps the project outcomes with academic and industry criteria, discusses ethical considerations, and examines the sustainability of the solution.

### Impact on Society, Environment and Sustainability

This section examines the broader effects of deploying secure web server technologies on both individuals and communities, while also discussing ethical considerations and long-term sustainability. The project, while rooted in technical implementation, serves as a case study for how secure digital infrastructures can improve quality of life, promote responsible computing practices, and contribute to a more sustainable digital ecosystem.

#### Impact on Life

The implementation of secure web servers has a direct impact on individuals’ daily digital interactions:

**Improved User Trust and Safety:**

Secure web servers ensure that data transmitted over the internet remains confidential and unaltered. This protection boosts user confidence in online transactions, information sharing, and communication systems.

**Enhanced Access to Services:**

By providing reliable and secure platforms, these technologies help ensure that essential services—such as online banking, telemedicine, and e-learning—are available without disruption. This ultimately contributes to improved quality of life and convenience for end users.

**Personal Data Protection:**

With growing concerns over data breaches and privacy invasion, implementing secure server configurations ensures that individuals’ sensitive information is less vulnerable to cyberattacks, fostering a safer digital environment.

#### Impact on Society & Environment

The ripple effect of deploying secure and efficient web infrastructure extends to society and the environment in several ways:

**Societal Benefits:**

A robust and secure digital infrastructure supports economic growth by enabling safer e-commerce, fostering innovation, and reducing cybercrime. Societies benefit when citizens engage with secure online platforms that protect personal and financial information.

**Reduction in Digital Divide:**

As secure and reliable digital services become more widespread, more individuals and communities gain access to essential online services. This inclusivity is crucial for educational, economic, and social opportunities, particularly in underserved regions.

**Environmental Impact:**

The use of efficient server configurations helps reduce energy consumption. A lightweight, well-optimized web server minimizes resource usage—lowering power consumption and reducing the carbon footprint of digital operations. As sustainable computing practices gain traction, environmentally responsible server designs contribute to energy efficiency and reduced electronic waste.

#### Ethical Aspects

Implementing secure server systems requires careful ethical considerations:

**Data Privacy and Integrity:**

Ensuring the protection of user data is paramount. The project emphasizes the ethical responsibility of developers to implement systems that defend against unauthorized data access, thereby upholding the rights of individuals.

**Transparency and Trust:**

Ethical practices in deploying secure web servers involve transparent communication about data protection measures with users. Providing clear information about the security protocols in place builds trust and empowers individuals to make informed choices about their digital interactions.

**Digital Inclusion:**

Ethically, developers must strive to create systems that are accessible and fair, ensuring that security measures do not inadvertently exclude certain groups. Implementing robust yet user-friendly security solutions balances protection with accessibility.

#### Sustainability Plan

A sustainable approach to secure web infrastructure is critical for long-term viability:

**Upgradability and Scalability:**

The project design emphasizes modular configurations (such as the use of virtual hosts) that allow for incremental updates. Transitioning from self-signed certificates to CA-signed certificates (e.g., via Let’s Encrypt) in production environments illustrates the pathway to a more secure and scalable deployment.

**Energy-Efficient Technologies:**

By optimizing server configurations to consume minimal resources, the project contributes to reducing power usage. Future iterations should focus on incorporating further energy-efficient measures, such as utilizing renewable energy sources for hosting data centers.

**Continuous Monitoring and Maintenance:**

Sustainable systems require ongoing maintenance. Regular updates, security patches, and performance monitoring ensure that the web server remains robust over time. Establishing routine audits and leveraging automated tools can help maintain system integrity and prolong the system's lifespan.

**Community and Open-Source Collaboration:**

The project leverages widely supported open-source technologies, which promotes community collaboration and shared innovation. Sustainable practices are strengthened when developers and organizations contribute to and benefit from a collaborative ecosystem, ensuring continuous improvement and knowledge sharing.

### Project Management and Team Work

This project was executed entirely by a single individual, requiring disciplined project management and effective self-organization. Despite working solo, a structured approach was adopted to ensure that every stage of the project—from planning through implementation and testing—was executed systematically.

**Planning and Scheduling:**

A clear project plan was developed at the outset, outlining specific objectives and deliverables such as Apache installation, virtual host configuration, and SSL setup. A timeline with milestones was created to manage the phases of work, ensuring that each task had a set deadline. This planning phase provided a roadmap that helped maintain focus and meet project goals.

**Task Management and Documentation:**

Each task was approached methodically, with detailed documentation maintained throughout the project lifecycle. A project diary was used to record challenges, solutions, and lessons learned. This log not only assisted in troubleshooting during development but also served as a valuable resource for future reference.

### Complex Engineering Problem

Addressing the intricate configuration of Apache and SSL modules required a deep understanding of networking and Linux file systems.

The project involved resolving common issues such as certificate errors and virtual host misconfigurations.

#### Mapping of Program Outcome

The project of setting up a secure Apache web server on Ubuntu addresses several core competencies that align with the intended Program Outcomes (PO’s). The following table justifies how the project’s design, implementation, and troubleshooting activities contribute to the attainment of each specified outcome.

Table 4.1: Justification of Program Outcomes

|  |  |
| --- | --- |
| **PO’s** | **Justification** |
| PO1 | The project demonstrates a solid foundation in managing operating systems and networking. The installation and configuration of the Apache web server, setting up virtual hosts, and basic command-line operations validate the understanding of core technical principles, which is essential for PO1. |
| PO2 | This project integrates diverse technologies such as Apache, OpenSSL, and virtual hosting. It requires practical application of engineering skills to generate and configure SSL certificates, troubleshoot server errors, and balance security with performance. These activities showcase the ability to apply theoretical concepts to solve real-world security challenges, justifying the attainment of PO2. |
| PO3 | The challenges faced—ranging from debugging configuration issues to optimizing server performance—demanded self-directed learning and iterative problem-solving. The project’s independent management and thorough documentation reflect the developer’s commitment to continuous improvement, aligning with the objectives of PO3 |

#### Complex Problem Solving

In this section, we analyze how the project tackled a complex engineering problem by mapping various problem-solving categories. The mapping illustrates how different aspects of the project—ranging from technical knowledge to system interdependence—were addressed. Table 4.2 summarizes the mapping, and the subsequent subsections provide a rationale for each category.

EP1: Depth of Knowledge Used strong OS, Apache & SSL config knowledge.

EP2: Conflicting Requirements Balanced between system security and performance.

EP3: Depth of Analysis Solved errors through log analysis & testing.

EP4: Familiarity of Issues Knew common setup issues, helped fast troubleshooting.

EP5: Applicable Codes/Standards Followed Apache & SSL best practices and configs.

EP6: Stakeholder Involvement Single project, but used online help and community feedback.

EP7: Inter-dependence Integrated OS, Apache, and SSL configs smoothly.

Chapter 4. Engineering Standards and Mapping

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EP1**  Dept of Knowledge | **EP2**  Range of Conflicting Requirements | **EP3**  Depth of Analysis | **EP4**  Familiarity of Issues | **EP5**  Extent of Applicable Codes | **EP6**  Extent  Of Stakeholder Involvement | **EP7**  Inter- dependence |
| IMG_256 | IMG_256 | IMG_256 | IMG_256 | IMG_256 | IMG_256 | IMG_256 |

#### Engineering Activities

The project involved various engineering activities ranging from resource usage to innovation. The following table shows the mapping of activities along with a short rationale.

EA1: Range of Resources Used OS tools, Apache modules, SSL libraries, domain configuration tools, etc.

EA2: Level of Interaction Required interaction with system files, services, ports, and server processes.

EA3: Innovation Applied manual SSL configuration and custom virtual host setup for local development.

EA4: Consequences on Society/Env Promotes awareness in secure local hosting and environmentally low-impact digital solutions.

EA5: Familiarity Familiar tools and platforms were used (Linux, Apache, OpenSSL), easing the implementation.

Table 4.3: Mapping with complex engineering activities.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EA1**  Range of resources | **EA2**  Level of Interactio | **EA3**  Innovation | **EA4**  Consequences for society and  environment | **EA5**  Familiarity |
| IMG_256 | IMG_256 | IMG_256 | IMG_256 | IMG_256 |

**Chapter 5**

# Conclusion

This final chapter summarizes the project outcomes, reflects on the lessons learned, identifies current limitations, and suggests possible directions for future work

### Summary

This project successfully demonstrated the deployment of a local Apache web server with virtual hosting and SSL integration using a custom domain. The setup included key configurations such as enabling Apache modules, creating virtual host files, setting up the /etc/hosts file, generating a self-signed SSL certificate, and linking it to the server. Despite being a local environment, the project accurately mimicked real-world server deployment and security practices, making it a valuable learning experience in Linux-based web hosting and secure communications.

### Limitation

While the project achieved its goals, a few limitations exist. The use of a self-signed SSL certificate means that web browsers mark the connection as "Not Secure" since the certificate isn't verified by a trusted Certificate Authority (CA). Additionally, the domain (diunextdoor.com) is configured only in the local environment through the /etc/hosts file and cannot be accessed from the public internet. Finally, the entire setup process is manual, which may pose challenges for those unfamiliar with Linux commands and server configuration.

### Future Work

To enhance the system, several improvements can be made. A major upgrade would be replacing the self-signed certificate with a trusted certificate from Let's Encrypt for real-world deployment. Hosting the website on a live server with DNS mapping would make it accessible globally. Furthermore, automating the configuration process through shell scripts or Ansible could make the setup easier and less error-prone, especially for beginners or educational purposes.

# References

[1] Apache HTTP Server Documentation

[2] Ubuntu Official Documentation

[3] OpenSSL Manual Pages

[4] DigitalOcean: How to Set Up Apache Virtual Hosts

[5] Mozilla SSL Configuration Guidelines