Project Report: AI-Powered Web Terminal

Domain

https://ai.shashithrashmika.tech

Technologies Used

Python (FastAPI), JavaScript (WebSockets), Google Gemini AI, Nginx, PM2, DigitalOcean (Ubuntu Linux)

1 Executive Summary (Simple Explanation)

The AI Web Terminal is a web application that allows users to control a Linux server using simple English sentences instead of complex terminal commands.

For example, instead of typing difficult Linux commands, users can type:

- "Compress this folder"
- "Find all Python files"

The AI (Google Gemini) understands the user's message and converts it into the correct Linux command. The server runs that command and sends the result back to the user in real time. This makes using a Linux terminal much easier for beginners.

2 System Architecture (How the System is Built)

This project follows a client-server model.

Main Parts

Frontend (Client)

- A web page that looks like a terminal.
- The user types commands here.
- It sends the input to the backend and displays the output.

Nginx (Reverse Proxy)

- Acts like a gatekeeper.
- Receives all traffic from users.
- Handles HTTPS (security).
- Sends requests to the Python backend.

Backend (FastAPI + Uvicorn)

- Main server application.
- Receives user input.
- Sends the input to the AI.
- Runs Linux commands and returns the output.

PM2 (Process Manager)

- Keeps the backend running all the time.
- Automatically restarts the server if it crashes.

AI (Google Gemini)

- Converts natural language into Linux commands.
- Example: "Show all files" → `ls`

3 Implementation (How It Works)

3.1 Backend (Python + FastAPI)

FastAPI is used because it supports real-time communication and provides high performance.

WebSockets

Normal HTTP opens and closes connections quickly. WebSockets keep the connection open, so the terminal can show output live, similar to a real terminal.

Command Execution

The server runs Linux commands using Python. It captures both standard output and error messages and sends them back to the user.

AI Integration

The AI is instructed to return only Linux commands without extra text. This ensures that the generated output can be safely executed by the system.

3.2 Frontend (JavaScript)

- The terminal UI is responsive and works on both mobile and desktop devices.
- The system automatically uses secure WebSocket (`wss`) when the site uses HTTPS.
- This avoids browser security issues.

4 Deployment & Server Setup

4.1 Cloud Server

- Hosted on DigitalOcean using Ubuntu Linux.
- A VPS is used to gain full control over the server environment.

4.2 PM2

- Keeps the backend running 24/7.
- Automatically restarts the application if it crashes.
- Helps with logging and monitoring.

4.3 Nginx

- Sits in front of the backend.
- Improves security.
- Handles HTTPS connections.
- Serves static files faster.

4.4 HTTPS Security (SSL Certificate)

- A free SSL certificate is used from Let's Encrypt.
- This secures the website using HTTPS.
- All traffic between the user and server is encrypted.